CA Enterprise Tester®

Application Guide

r3.1

This documentation (the "Documentation") and related computer software program (the "Software") (hereinafter collectively referred to as the "Product") is for the end user's informational purposes only and is subject to change or withdrawal by CA at any time.

This Product may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Product is confidential and proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of the Documentation for their own internal use, and may make one copy of the Software as reasonably required for back-up and disaster recovery purposes, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the provisions of the license for the Software are permitted to have access to such copies.

The right to print copies of the Documentation and to make a copy of the Software is limited to the period during which the license for the Product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to certify in writing to CA that all copies and partial copies of the Product have been returned to CA or destroyed.

EXCEPT AS OTHERWISE STATED IN THE APPLICABLE LICENSE AGREEMENT, TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS PRODUCT "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS PRODUCT, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

The use of this Product and any product referenced in the Documentation is governed by the end user's applicable license agreement.

The manufacturer of this Product is CA.

This Product is provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7013(c)(1)(ii), as applicable, or their successors.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Copyright © 2008 CA. All rights reserved.

CA Product References

This document references the following CA products:

- CA Enterprise Tester® (formerly know as ProTerm)
- CA FileAge®

Contact Technical Support

For online technical assistance and a complete list of locations, primary service hours, and telephone numbers, contact Technical Support at http://ca.com/support.

Contents

Chapter 1: Introduction

Introduction 1-1
CA Enterprise Tester Applications 1-2
CA Enterprise Tester Facilities 1-4
Unattended Batch Sessions 1-4
Multi-Session Management 1-5
Session Recordings and Traces 1-6
Automatic Regression Testing 1-6
Addressing CA Enterprise Tester from a REXX Exec 1-7
Application Program Interface 1-8
Release 3 Features 1-8
More Help 1-9
New Password Security 1-9
Light Pen and Cursor Select Support 1-10
3270 Special Character Support 1-10
3270 Terminal Read Support 1-11
Temporary VTAM Problem Recovery 1-11
System Sense Code Feedback 1-11
Context Sensitive PF Keys 1-12
Improved Interactive Dialogs 1-12
Long Wait Timeout Controls 1-13
Automatic User Libraries 1-13
Library Member Selection Lists 1-13
Packed Scripts 1-13
Packed Application Data 1-14
Dynamic Script Libraries 1-14
System Assigned DD Name Allocation 1-14
Script Initiated Session Capture 1-14
Persistent Script Console Messages 1-15
Flexible Script Logic Structures 1-15
Realistic Think Time Simulation 1-16
Recorded Session Documentation 1-16
Field Recording Options 1-17
Simplified Interactive Stress Tests 1-17
Uniform Multi-task Priority 1-18
Multi-port Log Sort Utility 1-18
Recorded REXX Execs 1-18

Attached REXX Environment Management 1-19
REXX Access Options 1-19
Virtual Session Display Options for REXX 1-20
Interactive Monitor Functions for REXX 1-20
The CA Enterprise Tester Command Monitor for REXX 1-21
PSL Logic in Trace 1-21
New Regression Testing Features and Options 1-22
Traced Session Recapture Utilities 1-24
Batch Job Setup Utilities 1-24
Release 3.1 Features 1-24
Release 3.0 Maintenance Incorporated 1-25
The Session Ager Utility 1-25
New PTEDATE pictures and pivot year control 1-25
New PSL DELAY statement options 1-25
Token Security Subroutine 1-25

Chapter 2: The Interactive System

CA Enterprise Tester Interactive System 2-1
CA Enterprise Tester Start-up and Control 2-3
Control Panel 2-4
Control Panel Commands 2-8
Starting Virtual Terminal Sessions 2-9
CA Enterprise Tester Session Panel 2-10
PF Keys and Commands 2-13
Keys Panel 2-14
Using the Keys Panel 2-16
Keys Panel Commands 2-17
Starting Sessions from a System Menu 2-18
CA Enterprise Tester System Menu 2-18
System Menu Commands 2-19
Using CA Enterprise Tester Sessions 2-20
Attention Identifier Keys 2-21
ISPF Dialog Restrictions 2-24
Ending a CA Enterprise Tester Session 2-26
Interactive Session Review 2-26
CA Enterprise Tester Session Review Panel 2-27
Session Review Panel Commands 2-28
Using Multiple Sessions 2-29
Recording Session Scripts 2-31
CA Enterprise Tester Record Panel 2-32
Record Panel Commands 2-35

Additional Record Commands . . . 2-35
Playing Session Scripts . . . 2-36
CA Enterprise Tester Play Panel . . . 2-37
Script Data Set Name . . . 2-38
EXEC PARM (Execute Parameters) . . . 2-39
Script Libraries . . . 2-40
Delay Percent Option . . . 2-40
Think Time Option . . . 2-41
Number of Ports Option . . . 2-41
PF Key Display . . . 2-41
Play Panel Commands . . . 2-41
CA Enterprise Tester Session Step Panel . . . 2-43
Session Step Panel Commands . . . 2-44
Status of a Played Script . . . 2-45
Free-Form Entry of PLAY Specifications . . . 2-45
Stopping a Script . . . 2-46
CA Enterprise Tester Session Wait Panel . . . 2-47
Session Wait Panel Commands . . . 2-48
Tracing Session Screens . . . 2-48
CA Enterprise Tester Trace Panel . . . 2-49
Data Set Name Parameters . . . 2-50
Pack Data Option . . . 2-51
Initial Caption Data Entry Area . . . 2-51
Protect Target Option . . . 2-51
Share PDS Option . . . 2-51
Trace Type Options . . . 2-52
Trace Data Options . . . 2-52
PF Key Display . . . 2-53
Trace Panel Commands . . . 2-53
Additional Trace Commands . . . 2-53
Sample CA Enterprise Tester Traces . . . 2-54
Browsing a Traced Session . . . 2-60
CA Enterprise Tester Browse Panel . . . 2-61
Data Set Name Parameters . . . 2-62
Responses Option . . . 2-62
Attributes Option . . . 2-63
Browse Panel Commands . . . 2-63
CA Enterprise Tester Session Browse Panel . . . 2-64
Session Browse Panel Commands . . . 2-65
Analyzing Traced Sessions . . . 2-66
CA Enterprise Tester Analyze Panel . . . 2-67
Trace Data Set and Member Name . . . 2-68
Script Library and Scripts . . . 2-68

Analyze Panel Commands 2-69
Interacting with Session Analyzers 2-71
Script Monitoring and Interactive Debug 2-71
CA Enterprise Tester Monitor Panel 2-72
Monitor Panel Commands 2-74
Script Logs and Diagnostic Traces 2-77
CA Enterprise Tester Log Panel 2-78
Log Panel Commands 2-81
Options and Controls 2-82
CA Enterprise Tester Options Panel 2-83
Option Panel Commands 2-87
Configuring Virtual Terminals 2-88
Interactive System SESSION Command 2-89
CA Enterprise Tester Terminal Panel 2-89
Terminal Panel Commands 2-91
VTAM Logon Mode Table 2-91

Chapter 3: Interactive System Command References

Interactive System Command Reference 3-1

Chapter 4: Password Security

Password Security 4-1
Token Security 4-2
CA Enterprise Tester Token Security Panel 4-5
Token Security Panel Commands 4-6
Maintaining Tokens and Passwords 4-7
Sharing User IDs and Tokens 4-8
Token Security Subroutine 4-8
Code Security 4-9
CA Enterprise Tester Code Security Panel 4-11
Code Security Panel Commands 4-12
Code Security Subroutines 4-13

Chapter 5: Script Language

The CA Enterprise Tester Script Language 5-2
Batch JCL 5-4
REGION Size 5-4
PARM - Passed Parameter Data Area 5-5

Data Set Definition Statements . . . 5-8
Script Language Overview . . . 5-9
Script Language Syntax . . . 5-10
CA Enterprise Tester Data Items . . . 5-16
Literal Values . . . 5-16
Actual Values . . . 5-17
Numbers . . . 5-18
Variable Fields . . . 5-21
Session Control . . . 5-26
Terminal Operation . . . 5-28
The Terminal Keyboard . . . 5-29
The Terminal Screen . . . 5-33
Structuring Script Code . . . 5-35
Compatibility Form IF Statement . . . 5-38
Variable Memory . . . 5-38
Character Variables . . . 5-39
Numeric Variables . . . 5-44
CA Enterprise Tester Defined Variables . . . 5-46
Data Set Access . . . 5-46
Processing Special Conditions and Errors . . . 5-49
Message Logging . . . 5-50
Operator Communication . . . 5-51
Session Traces and Recordings . . . 5-52
Script Language TRACE Statement . . . 5-53
Script Language RECORD Statement . . . 5-54
Condition Handling and Diagnostics Aids . . . 5-54
Handling Unusual Session Events and Errors . . . 5-55
Controlling Script Loops . . . 5-57
Forcing a Script ABEND . . . 5-57
Requesting a Dump . . . 5-57
Session Flows . . . 5-57
Bracket Protocol . . . 5-58
Flow Control . . . 5-61
Passed and Queued Sessions . . . 5-66

Chapter 6: PSL Reference

PSL Reference . . . 6-3
ABEND . . . 6-3
ALLOCATE . . . 6-4
AND . . . 6-6
CALL . . . 6-6

CLOSE 6-8
COMPUTE 6-9
What a COMPUTE Expression Contains 6-10
DATE 6-13
Formatting and Calculating Dates 6-13
DELAY 6-16
DELETE 6-18
DO 6-19
DROP 6-25
DUMP 6-26
ELSE 6-27
ENCODE 6-27
END 6-27
ENDDO 6-28
ENDIF 6-28
ENDRECORD 6-28
ENDTRACE 6-29
EXECUTE 6-29
EXIT 6-30
FLOW 6-30
FREE 6-31
GOTO 6-32
HANDLE 6-32
CONTENTION 6-33
EVENTS 6-34
ILLOGIC 6-35
LOGONFAIL 6-36
OUTBOUND 6-37
RECONNECT 6-38
RESPONSES 6-39
SESSIONFAIL 6-42
TIMEOUT 6-43
TURNAROUND 6-44
IF 6-45
INVITE 6-50
LIMIT 6-52
LINK 6-52
LOG 6-54
LOGOFF 6-55
LOGON 6-55
MASK 6-58
NEXT 6-61
OPEN 6-62

OR 6-64
PAUSE 6-64
PERFORM 6-64
READ 6-65
RECORD 6-68
SET 6-72
Assign a Value to a Variable 6-79
Propagate Data in a Variable Substring 6-80
Propagate Data in a Variable Array 6-80
Concatenate a List of Data Items 6-81
Initialize, Increment or Decrement a Variable 6-82
Edit Decimal Data 6-82
Convert Binary Data to or from Decimal Format 6-83
Convert Packed Data to or from Decimal Format 6-84
Convert Zoned Decimal Data to or from Decimal Format 6-85
Convert Character Data to or from Decimal Format 6-86
Convert Data to or from Hexadecimal Format 6-87
Manipulate Bit Patterns 6-88
Change the Case of Alphabetic Data 6-89
Determine Length of a Field of Data 6-89
Edit a Virtual Terminal Screen 6-90
Determine Attributes of a Screen Location 6-90
Save the Executing Line Number of a Script 6-91
THEN 6-91
THINK 6-92
TIME 6-95
Generating Random Numbers 6-95
TIMEOUT 6-96
TRACE 6-97
TRACK 6-102
Non-display Fields and Security 6-105
Screen Image Sequence Numbers 6-105
Executable Code in a Trace Data Set 6-106
Reading Logical Records in a Trace Data Set 6-106
Adding Comments and Logic to a Trace Data Set 6-106
Information Returned To a Script in CA Enterprise Tester Variables 6-109
TYPE 6-110
Password Security 6-112
Token Security 6-112
Code Security 6-112
Special Function Keyboard Keys 6-113
Cursor Position 6-113
3270 Light Pen and Cursor Select Keys 6-113

Attention Identifier Keys 6-114
Virtual Terminal Keyboard State 6-114
TYPE OVERLAY Statement 6-116
Typing TRACKED Screen Data 6-116
UPDATE 6-117
WAIT 6-119
WRITE 6-119
WTO 6-121
WTOR 6-124

Chapter 7: PSL Variables

PSL Variables 7-1

Chapter 8: Testing

Introduction 8-1
Regression Testing 8-3
Application Tests and Test Data 8-4
The CA Enterprise Tester Session Analyzer 8-5
Session Analyzer Options 8-9
Analyzing Sessions Interactively 8-14
Analyzing Session Mismatches 8-16
Masking Dates, Times, and Other Data 8-20
Advanced Analysis Techniques 8-22
Analyzing Sessions in Batch 8-23
Multiple Session Testing 8-25
A Basic Multiple Session Test 8-26
Controlling Multiple Session Tests 8-28
Using Select Scripts to Manage Multi-Session Tests 8-30
Creating Scripts with Power and Flex 8-32
Using Loops to Repeat Transactions 8-33
Unattended Multiple Session Tests 8-37

Chapter 9: Utilities

CA Enterprise Tester Utilities 9-1
The Utility Menu 9-1
Utility Options 9-2
The Session Ager 9-3
The Trace Utility 9-17

The Record Utility . . . 9-19
The Batch Play Utility . . . 9-23
The Multi-Port Select Utility . . . 9-25
The Multi-Port Watch Utility . . . 9-30
The Batch Compare Utility . . . 9-36
Batch Compare Utility Services . . . 9-37
The CA Enterprise Tester Job Setup Panel . . . 9-39
The CA Enterprise Tester Command Monitor for REXX . . . 9-41

Chapter 10: Commands for REXX

The Session Commands . . . 10-2
Tracing Exec-Driven Sessions . . . 10-4
CA Enterprise Tester Ports and Multiple Sessions . . . 10-5
Sharing Sessions with Called Execs . . . 10-7
Attaching Asynchronous Exec Tasks . . . 10-8
Queues and Intertask Communication . . . 10-10
Sharing Resources with Attached Execs . . . 10-15
Smart Exec-Driven Sessions . . . 10-16
ISPF Session Panels . . . 10-17
Monitoring Execs and Sessions . . . 10-19
The CA Enterprise Tester Command Monitor for REXX . . . 10-23
Command Monitor Request Panel Services . . . 10-23
Command Monitor Options . . . 10-25

Chapter 11: REXX Reference

CA Enterprise Tester REXX Reference . . . 11-2
Syntax Notation Format . . . 11-2
ACCESS . . . 11-2
Conditions and Return Codes . . . 11-4
ATTACH . . . 11-4
REXX Messages Management Options: . . . 11-7
Conditions and Return Codes . . . 11-8
CANCEL . . . 11-8
Conditions and Return Codes . . . 11-9
DELAY . . . 11-9
Conditions and Return Codes . . . 11-11
DELETEQ . . . 11-11
Conditions and Return Codes . . . 11-12
ENDTRACE . . . 11-12
Conditions and Return Codes . . . 11-12

HANDLE 11-13
Conditions and Return Codes 11-14
INVITE 11-14
Conditions and Return Codes: 11-16
LOCK 11-16
Conditions and Return Codes 11-17
LOGOFF 11-17
Conditions and Return Codes 11-18
LOGON 11-18
Conditions and Return Codes 11-22
MONITOR 11-23
Conditions and Return Codes 11-26
The Monitor Exec Interface 11-26
The Distributed Monitor Exec 11-27
POST 11-30
PULL 11-31
Conditions and Return Codes 11-32
PUSH 11-32
QUERY 11-33
QUEUE 11-35
SIGNAL 11-36
TRACE 11-38
TYPE 11-41
Conditions and Return Codes 11-44
UNLOCK 11-45
Conditions and Return Codes 11-45
WAIT 11-46
Conditions and Return Codes 11-47

Chapter 12: REXX Variables

CA Enterprise Tester REXX Variables 12-1

Chapter 13: Access Method

CA Enterprise Tester Access Method 13-1
Terminal Common Area 13-2
The LOGON CALL 13-4
The CONVERSE CALL 13-6
CA Enterprise Tester Screen Format 13-8
The INVITE CALL 13-9
The LOGOFF CALL 13-10

The WAIT CALL 13-10
Access Method Program Sample 13-11

Chapter 14: Logical Unit Codes

3270 Attribute Values 14-1
CA Enterprise Tester Logical Unit States 14-6
CA Enterprise Tester Session Status Flags 14-9
SNA Request Header (RH) Codes 14-10
SNA Request and Response Units (RU) 14-11
VTAM Return Codes and Feedback 14-14
SNA Sense Codes 14-15

Appendix A: Sample Scripts

Appendix A. Sample CA Enterprise Tester Scripts A-1
Closing CICS Data Sets from a Batch Job Step A-1
Opening CICS Data Sets from a Batch Job Step A-3
An Installation/Verification Job Manager A-5
Identifying Systems with Common Characteristics A-9
Creating Multiple Files From a Single File A-14
The CA Enterprise Tester Session Analyzer A-19
Batch Multi-Port Applications A-57
PTEMPLAY A-57
A Central Encoded Password Maintenance System A-66

Appendix B: Messages

Messages B-1

Chapter 1: Introduction

This chapter provides an overview of the capabilities and features of CA Enterprise Tester and gives some examples of the wide variety of problems that may be solved using the product.

- Introduction
- CA Enterprise Tester applications
- CA Enterprise Tester facilities
- Release 3 features
- Release 3.1 features

CA Enterprise Tester was previously known as ProTerm. Some of the application screens and functions continue to use the previous product name.

Introduction

CA Enterprise Tester is a general-purpose virtual terminal. A virtual terminal consists entirely of computer programs that run in an z/OS mainframe environment. Using virtual terminals, a user can obtain a virtual number of sessions with one or more online systems. A batch job can use a virtual terminal to log on to an online system and invoke any online software that was designed for use by a person using a real 3270-family terminal. To an online system like CICS or TSO, a CA Enterprise Tester virtual terminal appears to be just another 3270-family CRT.

CA Enterprise Tester provides the CA Enterprise Tester Script Language, a Batch System, an Interactive System, a Communication System, and a Host Command Environment for REXX execs. These facilities enhance and exploit the capabilities of CA Enterprise Tester virtual terminals.

The CA Enterprise Tester Script Language puts a programmed operator at the keyboard of a virtual terminal. CA Enterprise Tester Script Language may be used to write CA Enterprise Tester Scripts. The Script Language TYPE statement tells the programmed operator what to key into a virtual terminal. The Script Language IF statement may instruct the programmed operator to look at a virtual terminal screen. The CA Enterprise Tester Script Language can be used to create intelligent session scripts that automate any real terminal operator procedure.

The CA Enterprise Tester Batch System provides the capability to execute scripts from a batch address space, giving access to online system functions from unattended batch jobs. CA Enterprise Tester Scripts can also be executed under the control of the CA Enterprise Tester Interactive System.

The CA Enterprise Tester Interactive System runs under TSO and ISPF. The Interactive System provides a terminal user with a virtual number of terminal sessions and provides powerful and unique features that give CA Enterprise Tester sessions capabilities that are beyond those of ordinary terminal sessions and sessions provided by other multi-session managers. For example, CA Enterprise Tester can be instructed to record the activity that occurs on a terminal session. Recorded or user-written CA Enterprise Tester Scripts can be replayed on command. A replayed session can be automatically and flexibly compared to a trace of a former session to ensure that responses have not changed. Through the use of multiple virtual terminals, CA Enterprise Tester's regression testing capabilities may be used to perform complex application integration testing and online system stress testing.

The CA Enterprise Tester Communication System includes terminal emulation and communication logic that allows CA Enterprise Tester to dynamically connect to and communicate with online systems that support sessions with 3270 compatible terminals. The Interactive System and the Batch System use the Communication System. CA Enterprise Tester also provides an Access Method that allows the Communication System and virtual terminal to be used by a program that is written in any language that supports standard z/OS CALL conventions.

The CA Enterprise Tester Host Command Environment allows REXX execs to ADDRESS PROTERM Tester and to issue CA Enterprise Tester host commands.

CA Enterprise Tester's built in security features interface with security systems used in the z/OS environment, including RACF, ACF2, and CA Top Secret. Additionally, CA Enterprise Tester provides Security Exits that may be customized to control and automate local security requirements.

CA Enterprise Tester Applications

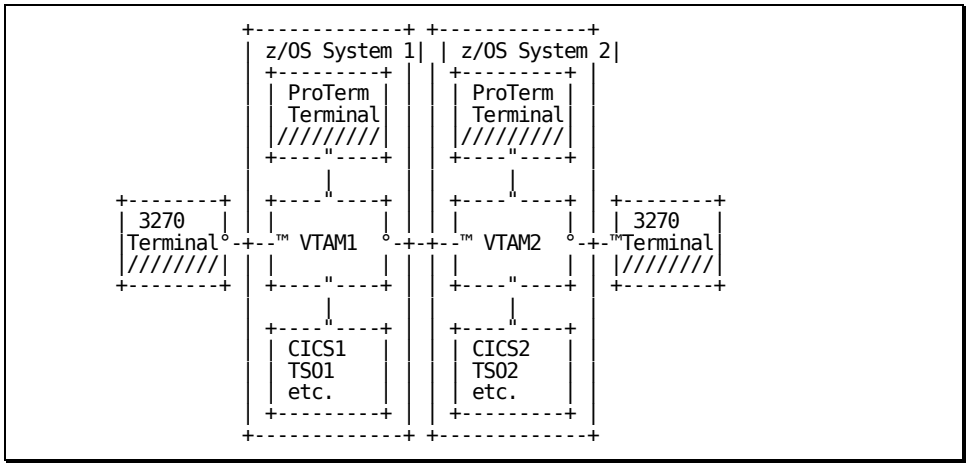
CA Enterprise Tester's general purpose capabilities provide cost effective and reliable solutions to many common business problems.

- Online System Control: Standard online control transactions can be invoked automatically by batch jobs to open or close online data sets or control other online resources, eliminating operator intervention, delay and error.
- Application System Control: Batch application systems may invoke online application start-of-day and end-of-day transactions during nightly offline file maintenance resulting in clean and reliable system interfaces.

- New Data Sharing Possibilities: A batch job can use existing online query transactions to extract data from online data sets for use by batch systems. Online system update transactions can be used by a batch job to perform mass loading or updating of online system data without the need to interfere with online data set availability.
- Eliminate Redundant Programming: Batch program access to existing online programs eliminates the need to write redundant batch versions of online update programs, saving programmer time and exploiting the investment in existing online programs.
- Online System Documentation: CA Enterprise Tester Screen Trace capabilities produce perfect printed or machine-readable screen images that can be easily incorporated into online system documentation and online system training material.
- Distributed Data Processing: Files uploaded to mainframe systems from distributed RJE systems and intelligent work stations can be automatically processed by simple CA Enterprise Tester Scripts that enter the data into mainframe systems using the same online system software that is used by conventional terminal network users.
- Multi-session Management: Provides preferred interactive environment multi-session capability to TSO/ISPF users. No session management overhead is incurred for a TSO/ ISPF user's primary terminal session. Multi-session management only products add overhead to all terminal traffic. CA Enterprise Tester provides access to TSO and CICS in split-screen mode.
- Application Regression Testing: Executable CA Enterprise Tester Scripts can be created automatically and be re-executed in batch or interactively. The Interactive System Monitor provides for complete control of regression tests and debugging of manually written CA Enterprise Tester Scripts.
- Application Integration Testing: Multiple CA Enterprise Tester Scripts can be executed asynchronously and concurrently to weed out deadlocks and other problems that may be identified when the components of a new application system are executed at the same time.
- Network Simulation: The Interactive System provides extensive multi-session global controls that support practical network simulation. Controls include the ability to stop and start multiple sessions that are each executing CA Enterprise Tester Session Scripts. The ability to vary the speed at which individual or all script-driven sessions operate allows you to find out when your system will break, by simply increasing the speed.

CA Enterprise Tester Facilities

CA Enterprise Tester's communication capabilities are made possible by a program that uses VTAM and SNA protocols to emulate the capabilities of 3270-compatible terminals. The program, referred to as a virtual terminal, executes in an z/OS environment. CA Enterprise Tester virtual terminals use VTAM to connect to online systems and networks in the same way that real 3270 terminals connect to online systems and networks. The following example illustrates the virtual terminals and 3270 terminals in an z/OS system.



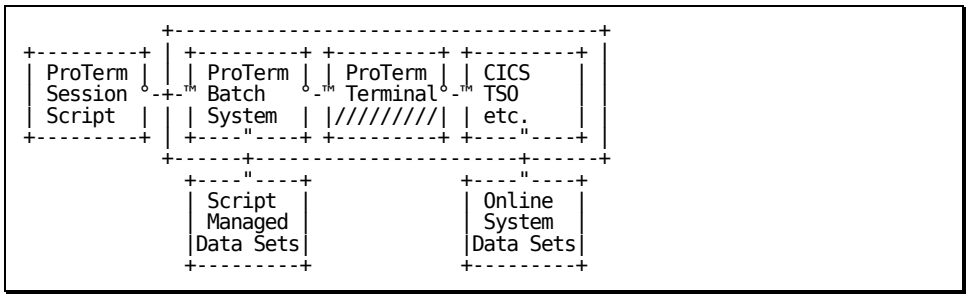
It is transparent to an online application that it is communicating with a CA Enterprise Tester virtual terminal rather than a real terminal. Any online system or application that was designed for use on a 3270 compatible terminal may be invoked using virtual terminals.

CA Enterprise Tester provides facilities that allow virtual terminal communication capabilities to be easily exploited.

Unattended Batch Sessions

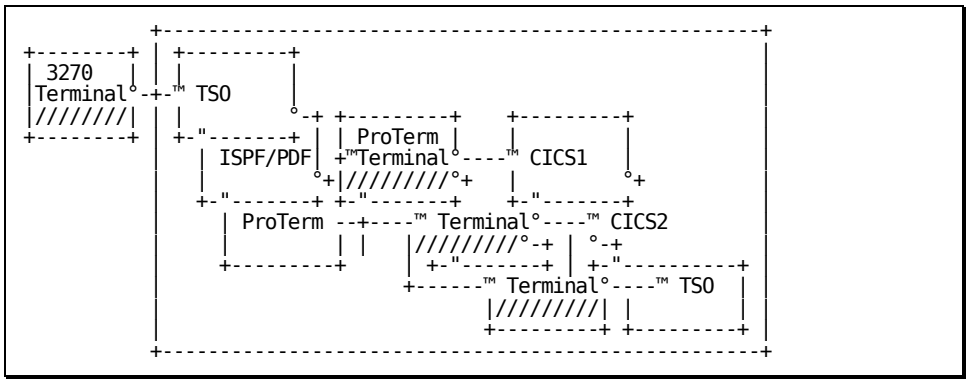
The CA Enterprise Tester Batch System allows z/OS batch jobs to exploit online system software. The Batch System defines a high-level Script Language. Scripts written using the CA Enterprise Tester Script Language are executed using the CA Enterprise Tester Script Language processor. Scripts tell CA Enterprise Tester what to key into the virtual terminal and what responses to look for. CA Enterprise Tester scripts can invoke any online system software that was designed for use by a person using a real terminal. Using the CA Enterprise Tester Script Language any manual terminal operator procedure can be easily and reliably automated.

A CA Enterprise Tester script may be executed as an z/OS batch job step or may be executed using the CA Enterprise Tester Interactive System Play panel. The following figure illustrates a CA Enterprise Tester script executing as a batch job step.



Multi-Session Management

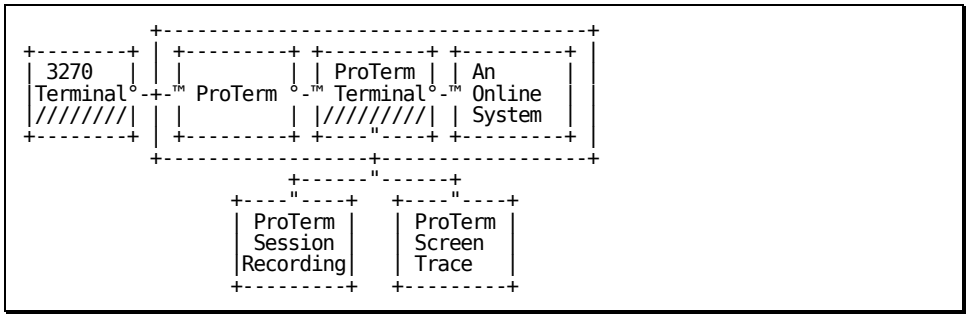
The CA Enterprise Tester Interactive System allows CA Enterprise Tester services to be invoked interactively under the control of TSO and ISPF/PDF. One feature of the CA Enterprise Tester Interactive System allows a terminal user to obtain a virtual number of concurrent interactive sessions with one or many online systems. The following figure illustrates the CA Enterprise Tester Interactive System multi-session environment.



The CA Enterprise Tester Session Manager allows the user's real keyboard to be used to control each virtual terminal session. Virtual terminal screens may be viewed in ISPF/PDF split-screen or TSO full-screen modes. Dozens of powerful features and commands support session switching, session review, PF key customization and many other capabilities.

Session Recordings and Traces

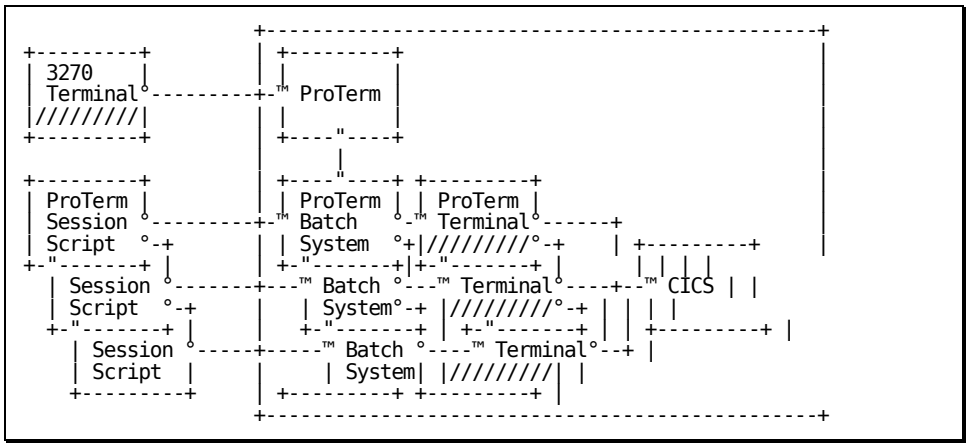
CA Enterprise Tester scripts can be created automatically. The CA Enterprise Tester Interactive System Record feature allows a CA Enterprise Tester Interactive System user to record virtual terminal keystrokes. The feature translates the terminal user's keystrokes into CA Enterprise Tester Script Language TYPE statements and stores these in a data set specified by the user. A recorded script may be executed using the CA Enterprise Tester Script Language processor and a virtual terminal to re-key terminal input as it was entered during the recorded CA Enterprise Tester session. The following example illustrates a CA Enterprise Tester session recording and screen trace.



In addition to the Interactive System Record feature, CA Enterprise Tester provides a separate screen trace feature. The screen trace feature allows session screen images to be saved at each virtual terminal input or output event. Screen traces may be used to capture the activity that occurs on CA Enterprise Tester Interactive or Batch System sessions. CA Enterprise Tester screen traces may be used for documentation or may be repeated using a CA Enterprise Tester script that re-keys traced data and compares the live response to the response read from the trace data set.

Automatic Regression Testing

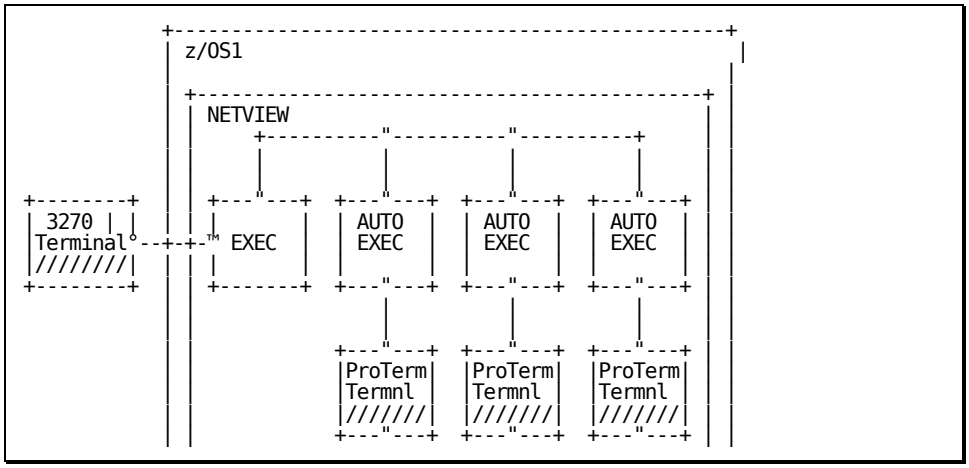
A recorded or user-written CA Enterprise Tester script may be executed under the control of the CA Enterprise Tester Interactive System. A script may be executed one session step or statement at a time, may be executed in the background freeing the TSO terminal to do other work, or may be executed using a movie mode that automatically displays inbound and outbound screens that flow on the session created by the script. The CA Enterprise Tester Interactive System may be used to execute many scripts concurrently. These capabilities, used in conjunction with interactive monitoring and debugging, and numerous commands and controls may be used to accomplish many types of regression testing, integration testing, online system stress testing and network simulation. The following example illustrates CA Enterprise Tester used in regression testing.

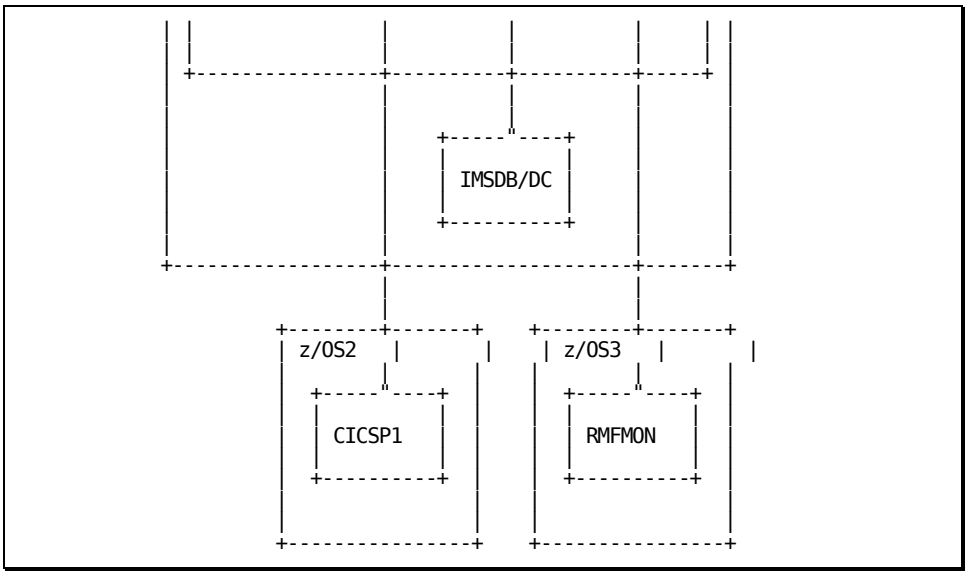


In the previous illustration, the CA Enterprise Tester Interactive System is used to concurrently execute several scripts for the purpose of testing the concurrent execution of several related online system application programs.

Addressing CA Enterprise Tester from a REXX Exec

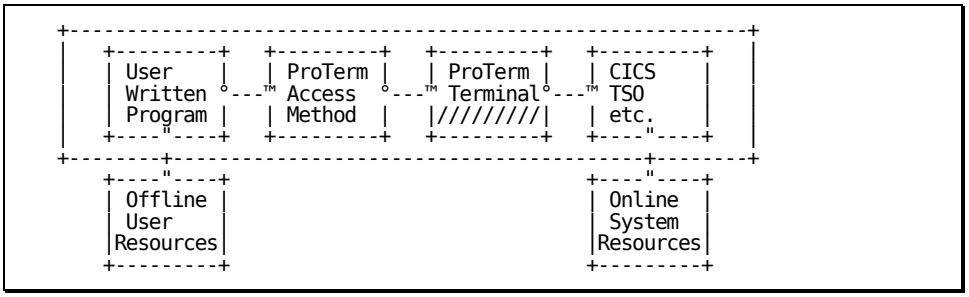
The CA Enterprise Tester Host Command Environment allows REXX execs to exploit all of the features of CA Enterprise Tester virtual terminals in a multi-session, synchronous or asynchronous environment. A REXX exec may ADDRESS PROTERM Tester and issue host commands to start, use and end CA Enterprise Tester virtual terminal sessions with online systems. A REXX exec may create multiple sessions on multiple ports and switch from session to session, causing events on the multiple sessions to occur serially or synchronously. REXX execs may use CA Enterprise Tester to attach other execs to run multiple sessions asynchronously. CA Enterprise Tester also provides host commands which enable communication among REXX execs using queues of data. The following example illustrates an application using the CA Enterprise Tester Host Command Environment and multiple REXX execs.





Application Program Interface

The CA Enterprise Tester Access Method allows COBOL and Assembler programs and other programs written in a language that supports standard z/OS CALL conventions, to use CA Enterprise Tester virtual terminals and communication capabilities. The following example illustrates a CA Enterprise Tester application program interface.



Release 3 Features

This section is a summary of new and improved CA Enterprise Tester Release 3.0 features. For complete information about each feature, refer to the applicable sections of this manual, or use CA Enterprise Tester Interactive System HELP to access information about CA Enterprise Tester online.

More Help

HELP now provides documentation on all CA Enterprise Tester functions. The HELP command or PF key now displays help for the function being used when help is requested. While using HELP, new help commands allow easy panel navigation and direct access to the help Table of Contents, Quick Reference lists, help Indexes, and other help topics.

- HELP commands, displayed on each help panel, display any page of the current topic, the next or previous topic, or the previous topic list.
- SLQR, ISQR and RXQR commands display Quick Reference lists for Script Language statements, Interactive System commands, and CA Enterprise Tester Commands for REXX.
- T and I commands display the Tutorial Table of Contents, and Tutorial Index.
- At index panels, any single letter used as a command displays an index of topics for the letter.
- ?keyword displays help for any turquoise keyword displayed at any index panel or other help panel.
- I;K;? Keyword displays help for any indexed verb, command or other CA Enterprise Tester Keyword directly from any help panel. For example, i;h;?help displays help for HELP from any help panel.

New Password Security

An all new password protection system, Token Security automatically replaces captured passwords with password tokens in captured session data, and stores encrypted passwords in private user data sets.

- Tokens are arbitrary symbols and provide no clue to the data they represent.
- Tokens can represent any confidential value.
- Tokens can be used like passwords in any script or session trace, but only to enter personal passwords.
- User-written CA Enterprise Tester scripts or REXX execs may use a new PSL TYPE statement or TYPE command keyword, TOKEN, to enter tokens instead of passwords.
- A token value may be changed at any time using a new SECURE command and the Token Security Panel.
- Scripts, execs and traced sessions that use password tokens instead of passwords are maintenance free when passwords are changed.

New Code Security options prevent or detect attempts to break the CA Enterprise Tester password encoding system.

New log security automatically blanks out passwords when raw 3270 request unit data is logged for debugging purposes.

Light Pen and Cursor Select Support

CA Enterprise Tester now provides complete support for light pen and cursor select key operations.

- Hardware light pens and cursor select keys now operate correctly when used with CA Enterprise Tester Zoom Mode sessions.
- A new PEN command simulates light pen and cursor select operations not supported by ISPF when CA Enterprise Tester sessions are used in Window Mode.
- Light pen and cursor select key operations are now captured automatically when CA Enterprise Tester sessions are traced or recorded.
- New SELECT, DESELECT and PEN keywords supported by the PSL TYPE statement and TYPE command for REXX, allow scripts and execs to simulate light pen and cursor select key operations using virtual terminals.

3270 Special Character Support

CA Enterprise Tester sessions, session recordings and traces, PSL statements and CA Enterprise Tester commands for REXX now support additional known and user-defined 3270 special keys and characters.

- 3270 Field Mark and DUP keys are now supported while using CA Enterprise Tester full screen sessions and extended attributes. Field Mark and DUP keys are now captured in traced or recorded session data, and may now be entered using the PSL TYPE statement or TYPE command for REXX.
- From time-to-time IBM defines or other vendors provide support for 3270 special characters in the hexadecimal 01 through 3F range that were not previously defined and were rejected by 3270 devices.

CA Enterprise Tester now supports additional 3270 special characters including line feed, escape and SUB (hexadecimal 25, 27 and 3F), that are now defined by IBM or supported by other vendor products.

In addition, a new system option allows 3270 special characters not yet defined to be defined by the customer, when they are defined by IBM or supported by other vendor products.

3270 Terminal Read Support

New Interactive System Terminal Options read a TSO display buffer to obtain information that is not automatically sent by a 3270 terminal when certain keys are used to send data. CA Enterprise Tester sessions with IMS MFS and other unusual 3270 applications that use 3270 read commands work correctly when the new options are used.

- A new Read-Buffer option uses the 3270 Read-Buffer command to copy a TSO terminal display buffer to a CA Enterprise Tester virtual terminal display buffer when data is entered into an unformatted terminal display, to capture nulls that are not automatically sent by a 3270 terminal.
- A new Read-Modified-All option uses the 3270 Read-Modified-All command to copy data from a TSO terminal display buffer to a CA Enterprise Tester virtual terminal display buffer when a light pen, cursor select key or PA key is used to send terminal input, to capture data that is not automatically sent by a 3270 terminal.

In addition, recorded TYPE statements now capture and rekey all data known to have been modified by a user, including modified data and data position obtained by reading the TSO terminal buffer. As a result, when the new terminal read options are used to capture a session with an application that issues 3270 read commands, the captured session also works correctly.

Temporary VTAM Problem Recovery

New VTAM error handling logic now automatically retries requests rejected by VTAM because of temporary storage shortages or other retrievable condition, as many times and as frequently as permitted by new CA Enterprise Tester system options, allowing CA Enterprise Tester applications to recover from temporary stress-related conditions and continue normal processing.

When CA Enterprise Tester scripts are used to drive a multi-session online system test, the new recovery logic allows scripts to survive temporary VTAM short on storage conditions that may occur if VTAM buffers are not tuned to handle a large number of concurrent session requests.

System Sense Code Feedback

If a negative response is received to a request sent by any PSL session statement or session command for REXX, the SNA sense code sent by VTAM or an online system indicating the reason for the negative response is now returned in a new script or REXX variable, PTERSENS.

Network management applications that access online systems to verify system availability may now analyze returned sense codes to determine the cause of network problems.

Context Sensitive PF Keys

All PF keys now enter user-assigned commands that are supported by the CA Enterprise Tester function being used. The command entered by each PF key while using a CA Enterprise Tester function is one of three commands assigned using the CA Enterprise Tester Keys Panel:

- ISPF Commands are active while using all CA Enterprise Tester panels, except when they are overridden by assigned Window Commands that are supported by the CA Enterprise Tester panel being used.
- Window Commands are CA Enterprise Tester commands that perform unique CA Enterprise Tester functions, for example, pass PF keys through to online systems, but are active only while using CA Enterprise Tester panels that support the assigned commands.
- Zoom Commands are CA Enterprise Tester commands and are always active while using full-screen Zoom Mode session displays.

Improved Interactive Dialogs

New alias command names, messages, and cursor control assist users in understanding and using CA Enterprise Tester Interactive System functions.

- The WAIT command now has a new alias name, VIEW, that better describes the purpose of the command; to view a script-driven session movie.
- New START and SELECT commands are new alias names for the SESSION command. All three names are abbreviated S and may be used to start a session, select an active session or to display the Interactive System Session Menu.
- All CA Enterprise Tester panels now display concise messages when invalid data sets are specified or library members are not found, and position the cursor at a field when a value must be entered or corrected.
- A new display option may be used to display condensed session movies and reviews that show only final online system responses, or to display each online system response as a separate session step when online systems send multiple consecutive responses to a request.

Long Wait Timeout Controls

z/OS system options allow an installation to limit the amount of time that a job may wait for the completion of an event and to cancel a job with an S522 abend if a job waits longer than the time limit. New CA Enterprise Tester options may be used to avoid S522 abends, allowing CA Enterprise Tester applications to wait longer than an installation specified time limit for a response from a long running transaction.

- A new PSL statement, TIMEOUT S522 OFF, or a new JCL parameter, S522OFF, may be used to avoid S522 abends when long running transactions are used by CA Enterprise Tester batch applications.
- A new Terminal Panel option, Disable TSO Timeout?, may be used to avoid TSO S522 abends when a script that uses long running transactions is played using the CA Enterprise Tester Interactive System.

Automatic User Libraries

CA Enterprise Tester script, trace, log, token and profile data sets may be created automatically for each CA Enterprise Tester user when first needed, using a data set naming convention, DCB information, space and other data set parameters specified by system options.

Library Member Selection Lists

All CA Enterprise Tester panels that support partitioned libraries now display member selection lists when a member name is required and not entered, or when a member name pattern is entered. While using a member selection list, select codes may be used to process listed members.

- B may be used to Browse a member.
- E may be used to Edit a member.
- S may be used to Select a member to be processed by the panel that was used to display the member selection list.

Packed Scripts

CA Enterprise Tester scripts stored in the ISPF packed data format may now be executed. New Record Panel options, Record statement keywords, batch JCL parameters, or system default options may be used to pack recorded scripts automatically when they are captured. Packed scripts may reduce the size of script libraries by up to 80%.

Packed Application Data

A new PSL OPEN statement keyword, UNPACK, may be specified when a data set with fixed length records up to 256 bytes long is opened for input, to instruct the READ statement to return unpacked logical records if the data set contains records written in ISPF packed data format.

CA Enterprise Tester session analyzers now use OPEN UNPACK to read trace lists, allowing trace lists to be stored in ISPF packed data format.

Dynamic Script Libraries

The PSL LINK statement now allocates script libraries automatically when a script links to another script and script libraries are not allocated or the script is not found in allocated libraries.

The new library allocation feature allows scripts to link to other scripts in the same library or to scripts in a central script library specified in CA Enterprise Tester system options, without specifying Play Panel libraries or PTELIB DD statements.

System Assigned DD Name Allocation

When a null-valued variable is used as a DD name, the PSL ALLOCATE statement now allocates the specified data set using a system-assigned DD name and returns the assigned DD name to the script in the specified DD name variable. The PSL READ statement and other statements may refer to the allocated data set using the returned DD name variable.

When a script that allocates a data set using a system-assigned DD name is played on multiple ports at the same time, a unique DD name is assigned to each script task when the data set is allocated.

Script Initiated Session Capture

By default, a session trace or recording started by a script using a TRACE TO or RECORD TO statement is automatically ended when the script ends. A TRACE TO or RECORD TO statement may now use a new keyword, PASS to leave the trace or recording started by the statement active when the script ends, to capture subsequent interactive session activity.

The new PASS option may be used to create procedures that start traces or recordings to capture user sessions without using CA Enterprise Tester panels.

Persistent Script Console Messages

By default, console messages issued using the PSL WTO statement are automatically deleted from the console when a script ends.

A new WTO statement keyword, NODOM, may now be used to issue console messages that are not deleted by CA Enterprise Tester, and that remain outstanding until they are deleted by a console operator or console options.

Flexible Script Logic Structures

New PSL DO statement keywords and related statements provide flexible options that support new script logic structures.

- New DO statement keywords, BEFORE and AFTER, allow a script author, and not language rules, to decide if a DO procedure is executed before or after WHILE or UNTIL conditions are evaluated.
- New ITERATE and LEAVE statements begin a new iteration of or end the current DO structure immediately, and may be used in any DO structure even if no DO conditions are specified.
- By default, an unconditional DO routine is executed one time. A new DO keyword, FOREVER, may be used to repeat a DO structure until a LEAVE statement or other statement ends or otherwise exits the structure.

An unconditional DO structure that contains IF-THEN-LEAVE-ENDIF structures performs exactly the same function as select statements supported by other languages. However, because logic in CA Enterprise Tester DO structures is not restricted to executing just one selected routine, DO structures are more flexible than conventional select structures.

Realistic Think Time Simulation

A new PSL DELAY statement may now be used in testing applications to create script-driven sessions that enter transactions at a realistic irregular rate, like people do.

- New session capture options may now be used to create DELAY statements automatically, that capture the elapsed time between each transaction response and the next terminal input.
- When a captured session is repeated, captured DELAY statements delay script execution prior to repeating each captured transaction and cause transactions to be entered at the same irregular rate as the original session.
- A new DELAY PERCENT statement causes subsequent DELAY statements to use a specified percent of DELAY statement time. DELAY PERCENT regulates transaction rate by compressing or expanding captured delay times proportionally.

Delays caused by the new DELAY statement or by the THINK statement, delay only the script end of a session, and not the online system. While a delay is in progress, any messages sent by an online system are received automatically.

The new logic takes advantage of delay time to compensate for slow response time and virtually eliminates synchronization problems when captured delays or adequate think time are used. Now, even sessions with online systems that send unpredictable immediate or delayed responses or that send an unpredictable number of responses to terminal requests, repeat reliably.

Recorded Session Documentation

New Record Panel options, RECORD statement keywords or default system options may be used to automatically record a partial virtual terminal screen image as comments preceding each recorded TYPE statement.

Recorded screen image comments visually document the flow of a recorded session and make it easy to follow the logic of a recorded script.

Field Recording Options

New Record Panel options, RECORD statement keywords, batch JCL parameters or default system options may be used to specify what fields are to be captured in recorded TYPE statements, either fields that were changed by the terminal user or all modifiable fields whose modified data tag (MDT) is set, even if the MDT was set by the online system and not by user input. Previous CA Enterprise Tester releases recorded fields changed by the user. Modified fields may be recorded to capture all fields that were entered by a terminal user, but were not changed because the entered data was already displayed and the MDT was already set.

Simplified Interactive Stress Tests

The Play Panel may now be used to start an entire multi-session online system test using only a single command.

- A new Play Panel option, Number of Ports, allows authorized users to play a script on a specified number of ports at the same time.
- Each copy of a script played on multiple ports runs as a separate z/OS task in the TSO address space in which a multi-port play request is entered.
- Simple user-written scripts or scripts created automatically using CA Enterprise Tester's new Multi-Port Select Utility, may be played on multiple ports to start multi-session tests using any combination of recorded or user-written scripts, by entering a single Play Panel Request.
- New Play Panel options, Delay Percent and Think Time, may be used to set the initial speed of all sessions created by scripts played at the same time.
- While a test started using a multi-port play request is in progress, CA Enterprise Tester panels may be used to monitor the progress of all script tasks and sessions or to monitor individual sessions selectively.
- While a test started using a multi-port play request is in progress, commands may be used to stop, start or regulate the speed of one or all script-driven sessions.

Uniform Multi-task Priority

When scripts are played on multiple ports at the same time, an z/OS task is used to execute each played script. Scripts running on multiple ports run asynchronously at the same z/OS dispatching priority and compete for system resources. However, because z/OS dispatches equal priority ready tasks in TCB chain order, when previous CA Enterprise Tester releases were used to play many scripts at the same time to drive a multi-session online system test, scripts that were started first tended to run faster.

A new multi-task priority feature causes script tasks started using the Interactive System Play command and exec tasks created using the CA Enterprise Tester ATTACH command for REXX, to relinquish priority after receiving online system responses. The new multi-task priority feature ensures uniform multi-task dispatch ability and makes CA Enterprise Tester scripts or REXX execs that run at the same time tend to run at the same speed, regardless of the order in which they were started.

Multi-port Log Sort Utility

A new Log Sort Utility sorts messages in a CA Enterprise Tester log into order by port and time. The Log Sort Utility may be used to sort a log captured while conducting a multi-session online system test into port order, allowing all chronological messages for each script task to be viewed separately.

Recorded REXX Execs

A CA Enterprise Tester session may now be recorded as either a CA Enterprise Tester script or as a REXX exec, at the option of the user. Except for required initial REXX comments and instructions, and REXX punctuation, a recorded REXX exec is identical to a recorded CA Enterprise Tester script, and like a CA Enterprise Tester script, creates a session similar to a captured session when executed. Recording options apply to both recorded CA Enterprise Tester scripts and REXX execs.

- New system options allow the customer to customize REXX text to be recorded when a REXX recording is started, including an initial /* REXX */ comment, an ADDRESS PROTERM Tester instruction, and any other desired initial comments or instructions.
- When a REXX exec is recorded, session initiation is captured as a CA Enterprise Tester LOGON command for REXX, user keystrokes are captured as TYPE commands, and INVITE commands are recorded to receive online system initiated transactions, for example, a sign on prompt sent by an online system when a session is started.

- New DELAY commands for REXX, that work like the new PSL DELAY statements, may be recorded to capture user think time.
- Virtual terminal screen rows may be recorded as REXX comments. Recorded screen image comments visually document the flow of a recorded session and make it easy to follow the logic of a recorded exec.

Attached REXX Environment Management

New ATTACH command parameters or new CA Enterprise Tester system options may now be used to manage REXX SAY destinations, REXX message destinations, and REXX libraries in REXX Environments created by the CA Enterprise Tester ATTACH command for REXX..

- New ATTACH command keywords, SAYDD and/or SAYDEST, may be used to specify the DD name and/or destination of REXX SAY messages in the attached REXX Environment. The message destination data set, either the TSO terminal or a JES SYSOUT class, is allocated automatically if the message DD is not already allocated.
- A new ATTACH command keyword, MSGDEST, may be used to send REXX error messages in the attached REXX Environment to the REXX SAY destination, via WTO, or to both destinations.
- A new ATTACH command keyword, EXECLIB, may be used to specify the DD name of the REXX exec libraries from which execs are to be loaded in the attached REXX environment.

REXX Access Options

New CA Enterprise Tester ACCESS command keywords may now be used by REXX multi-task or multi-session applications to select and control CA Enterprise Tester ports.

- EXECPORT may be used to access the current exec port.
- NEWPORT may be used to access an unused port.
- FREEPORT may be used reuse an idle port.
- LASTPORT may be used to identify and access a port that was accessed by a called or calling exec.
- LOCK may now be used to lock a port when it is accessed.

Virtual Session Display Options for REXX

New LOGON and QUERY SESSION command keywords, may now be used by REXX execs to choose the format of virtual terminal display buffers returned by CA Enterprise Tester session commands.

- PANELDATA formats virtual terminal display buffers for display at a terminal using an ISPF panel.
- IMAGEDATA returns printable display buffers that may be printed or saved for use in documentation.
- BASICDATA returns a single display buffer that contains CA Enterprise Tester basic data stream attribute values.
- EXTDSDATA returns a display buffer containing real 3270 basic field attribute values and an extended attribute buffer that contains extended attribute values.

Interactive Monitor Functions for REXX

A new CA Enterprise Tester MONITOR command for REXX may be used to monitor CA Enterprise Tester commands and to display information about selected events as they occur while an exec is executing.

- SIGNALS may be monitored to display error messages automatically.
- COMMANDS may be monitored to display CA Enterprise Tester commands at a terminal as they are executed by REXX execs.
- RESPONSES may be monitored to display online system responses to transactions entered by an exec using CA Enterprise Tester session commands.
- CONDITIONS may be used to display unusual CA Enterprise Tester conditions returned to REXX execs at the completion of CA Enterprise Tester commands.
- REXX variable values used or returned by CA Enterprise Tester commands may be displayed to debug an exec.

Debugging is only one application of the MONITOR command. Instead of just displaying information about detected events, a MONITOR command may be used to activate a user-written monitor exec to process detected events. Monitor execs may be used to develop general purpose execs that monitor other execs or exec-driven sessions. For example, the CA Enterprise Tester Command Monitor for REXX uses the MONITOR command and other commands to execute and monitor one or more other execs, and may be used to drive a multi-session online system test and to view multiple exec-driven sessions while the test is in progress.

The CA Enterprise Tester Command Monitor for REXX

A new Interactive System utility, the CA Enterprise Tester Command Monitor for REXX may be used to execute and monitor one or more REXX execs that use CA Enterprise Tester commands, and to display command events and exec-driven sessions while the exec or execs are executing.

- The Command Monitor may be used to execute an unmodified recorded REXX exec and to display the session created by the exec while the exec is executing.
- The Command Monitor may be used to execute multiple recorded or user-written REXX execs at the same time to drive and monitor a multi-session online system test.
- When multiple execs are executed, each exec is executed in a new attached REXX Environment and runs as an asynchronous z/OS task.
- Sessions driven by multiple attached execs are displayed in windows allowing multiple concurrent session events to be viewed at a single terminal at the same time.

PSL Logic in Trace

CA Enterprise Tester Script Language statements may now be mixed with virtual terminal screen images captured when a session is traced.

A new TRACK statement keyword, STATEMENT may now be used to read the next screen image or PSL statement found in a trace data set. When a PSL statement is found, the complete statement is returned in a new variable, PTETLINE, and may be executed using the PSL EXECUTE statement.

The distributed session analyzers now use TRACK STATEMENT to process trace data and execute PSL statements when they are returned by the TRACK statement.

PSL LOGON and DELAY statements may now be captured in a trace data set automatically when a CA Enterprise Tester session is traced.

- Captured LOGON statements capture any logon data used to start traced sessions, and use the data to start similar sessions when they are executed by session analyzers.
- Captured DELAY statements capture the actual user think time that precedes each transaction captured in a session trace, and simulate think time delays when they are executed by session analyzers.

The new support for PSL statements in trace data is general purpose. Users may now add PSL statements and comments to trace data for any logical reason, using ISPF EDIT.

- PSL comments may be inserted in trace data sets to include test documentation in traced session data.
- PSL /* and */ comment delimiters may be inserted before and after a sequence of traced screen images to skip commented-out transactions when the traced session is repeated to perform regression testing.
- PSL SET statements that set the variable, & mask to the name of a mask script may be inserted in a trace data set to automatically activate the script when the traced session is analyzed.

New Trace Panel options, batch JCL parameters, PSL TRACE statement and TRACE command keywords, or default system options may be used to capture PSL LOGON and DELAY statements and other optional trace data.

- A new trace option may be used to capture only online system responses that free the terminal keyboard to condense multi-message responses, or to capture a virtual terminal screen image when each message in a multi-message response is received.
- A new remembered Trace Panel option is now used to specify the format of traced virtual terminal screen images, either Format, Image or Upper case.

New Regression Testing Features and Options

A new distributed session analyzer, PTESA, performs all functions performed by previously distributed session analyzers, (PTEATI and PTEATB), and performs new functions that simplify and improve the quality of regression testing.

- The new session analyzer support PSL statements in traced session data. TRACK STATEMENT and EXECUTE are used to read and execute PSL statements found in a trace data set.
- When trace data begins with a LOGON statement, the Logons used to start the new session that is used to repeat the traced session.
- PTESA provides flexible timing options, including an option to use captured PSL DELAY statements, that repeat sessions reliably, even when repeating sessions with online systems that respond unpredictably.
- While PTESA provides complete support for trace lists and all trace list functions supported by previously distributed session analyzers, trace lists are no longer required to customize analyzer options. CA Enterprise Tester scripts may LINK to PTESA any number of times in a single run to repeat any number of partial sessions, complete sessions or trace lists, and may use the same or different analyzer options for each session.

- A new analyzer option, EDIT, may be used to activate a user-written script that may synchronize live and traced sessions, by repositioning a trace data set and/or a live session automatically when working with systems that send messages at unpredictable points in a session. For example, a distributed sample EDIT script, PTETSOED, may be used to process notify messages found in a trace data set or sent to a TSO user at the completion of a batch job submitted while repeating a traced TSO session.
- A new analyzer option, TYPE, may be used to activate a user-written script that may key in any desired data after traced screen data has been rekeyed and before data is sent to an online system. For example, if an online application performs a desired function only when the current date is entered, a TYPE script may be used to automatically key the current date over the obsolete date rekeyed from a traced screen image, while analyzing any traced session that uses the application.
- A new analyzer option, CASE, may be used to ignore differences in alphabetic case, and allows traced sessions with transactions that previously displayed only upper case characters to be used to perform regression testing when the transactions are modified to display dual case alphabetic characters.

  When a new CA Enterprise Tester system option is used to customize CA Enterprise Tester alphabetic case translation tables, the CASE option works correctly with any international single-byte alphabet.

In addition, PTESA provides improved test results documentation that includes uniform information whether regression is performed interactively or automatically using batch jobs.

- PTESA mismatch reports include the full screen image that was input to a transaction and old and new full screen images of responses that did not match. In addition, each pair of screen rows that do not match are reported separately and each mismatching byte of each row is flagged, making it easier to identify and understand every thing that changed when response mismatches are detected.
- A new screen image sequence number, captured in the border of each virtual terminal screen image when a session is traced, and returned by the PSL TRACK statement, is now included in mismatch messages, and may be used as a search argument to find a particular screen image when a trace data set is browsed or edited using ISPF, or to locate a particular screen image when a session is viewed using the CA Enterprise Tester Browse Panel.

Traced Session Recapture Utilities

It is no longer necessary to repeat a session captured in one form to capture the session in another form. New interactive utilities print, trace or record a session, using an existing format trace data set as input instead of a live session.

- A new Trace Utility reads a session trace data set and prints or creates a new session trace data set using specified trace options. The Trace Utility may be used to store printable screen images for use in documentation, to add DELAY statements to a trace captured without delays, or to remove unwanted PSL statements or screen images.
- A new Record Utility reads a session trace data set and records a CA Enterprise Tester script or REXX exec that creates a session similar to the traced session when executed. The Record Utility supports options similar to the options available when a live session is recorded.

Batch Job Setup Utilities

New interactive utilities may be used to set up JCL and control statements to run CA Enterprise Tester batch applications.

- A new Batch Play Utility sets up required and optional JCL defined by CA Enterprise Tester, to play any CA Enterprise Tester script in batch, and displays the generated JCL using ISPF EDIT, allowing the JCL to be modified, saved or submitted as desired.
- A new Multi-Port Watch Utility sets up a job that uses TSO, the CA Enterprise Tester Interactive System, and scripts created using the new Multi-Port Select Utility, to start and watch a multi-session test in exactly the same way that a person would start and watch the test interactively. Jobs created using the Watch Utility may be scheduled to run at night, to perform resource-intensive testing automatically when system resources are more available.
- A new Batch Compare Utility sets up a job to play a script, PTECTTB, that compares two previously captured trace data sets to identify differences. PTECTTB uses mask scripts and reports mismatches using techniques similar to session analyzers.

Release 3.1 Features

This section provides a summary of new and improved CA Enterprise Tester Release 3.1 features.

Release 3.0 Maintenance Incorporated

Corrections to all Release 3.0 product problems reported as of the date of Release 3.1 general availability are incorporated in the Release 3.1 product materials.

The Session Ager Utility

The Session Ager is used to age date values in session data that was previously captured using the CA Enterprise Tester Trace facility. Using aged test sessions created using the Session Ager, aged test data created using the CA-FileAge product, and CA Enterprise Tester session analyzers, online software tests performed manually using a particular current system date to verify date-sensitive applications can be repeated and analyzed automatically using other current system dates. For complete information on the Session Ager utility, refer to the Chapter "Utilities."

New PTEDATE pictures and pivot year control

The distributed date logic script, PTEDATE, now supports additional date picture mnemonics, zm, zd, mn and win, and also provides an option to specify a pivot year to be used to determine the century of dates with 2-digit years. For details, refer to the section DATE on the chapter "PSL Reference."

New PSL DELAY statement options

The PSL DELAY statement now supports new keywords, INPUT and BRACKET, that may be used to specify when DELAY statements are to cause actual session delays. When DELAY BRACKET is specified, session delays are avoided except in the case that it is possible to receive an anticipated message from the online system. The distributed session analyzer, PTESA, incorporates the DELAY BRACKET option when permitted in order to maximize the speed of analyzed sessions.

Token Security Subroutine

A new token maintenance subroutine may now be called by any program that supports standard z/OS call conventions to assign a new value to a named token that belongs to the user ID that owns the address space in which the token maintenance routine is called. The new token maintenance routine allows token values to be maintained by user written programs as an alternative to using the CA Enterprise Tester Token Security Panel to maintain token values manually when user passwords are changed.

Chapter 2: The Interactive System

This chapter describes how to use CA Enterprise Tester interactive functions. The Interactive System allows TSO/ISPF users to start sessions with other VTAM online systems and provides the ability to capture and replay sessions or to play scripts that generate original sessions. These features may be used to automate regression testing, quality assurance testing, application system integration testing and to perform network simulations. The ability to play intelligent user written session scripts interactively provides new ways to accomplish volume terminal tasks, and boosts productivity by eliminating tedium and expensive ado programming.

- CA Enterprise Tester Interactive System
- CA Enterprise Tester Start-up and Control
- Starting Virtual Terminal Sessions
- PF Keys and Commands
- Starting Sessions from a System Menu
- Using CA Enterprise Tester Sessions
- Interactive Session Review
- Using Multiple Sessions
- Recording Session Scripts
- CA Enterprise Tester Record Panel
- Playing Session Scripts
- Tracing Session Screens
- Browsing a Traced Session
- Analyzing Traced Sessions
- Script Monitoring and Interactive Debug
- Script Logs and Diagnostic Traces
- Options and Controls
- Configuring Virtual Terminals

CA Enterprise Tester Interactive System

The CA Enterprise Tester Interactive System operates under the control of TSO and ISPF and provides a user with a variety of session related capabilities.

Multiple sessions may be obtained with one or many online systems while retaining the user's primary TSO session. CA Enterprise Tester sessions have capabilities not provided by ordinary terminal sessions. CA Enterprise Tester sessions can be viewed and used in ISPF split-screen mode or TSO full-screen mode. REVIEW and STEP commands support reviewing of recent session screens.

The Interactive System Session Record Facility monitors activity on a CA Enterprise Tester session and automatically generates a CA Enterprise Tester script. A generated script consists of CA Enterprise Tester Script Language LOGON, TYPE, INVITE, LOGOFF and, optionally, DELAY statements. When a recorded script is executed, a new session is started and captured keystrokes are automatically re-entered to repeat the captured session.

The Interactive System Play Facility allows a CA Enterprise Tester script to be executed online and to drive Interactive System sessions with other online systems. Scripts may be played to accomplish many types of online system testing or may be played to enhance terminal user productivity by eliminating repetitive terminal tasks.

The Interactive System Session Trace Facility monitors activity on a CA Enterprise Tester session and captures session events in the form of screen images that reflect terminal inputs and online system responses. Optional trace formats allow a session trace to be used for documentation or as input to a CA Enterprise Tester session analyzer.

The Interactive System Browse Facility may be used to automatically create a movie of a previous session with an online system which was captured using the Trace Facility. The session will be reproduced using the screen images from the trace data set without actually creating a live online system session. Session events may be reproduced for demonstration purposes without affecting the operation of the online system or the data accessed by the system.

The Interactive System Analyze Facility may be used to automate regression testing of online applications. A CA Enterprise Tester session analyzer may be used interactively to automatically re-create a previously traced session. A session analyzer re-keys captured keystrokes, and when an online system responds to the terminal input, the session analyzer compares the response to the previously captured response and reports differences to the terminal user.

The Interactive System Monitor Facility provides interactive monitoring and debugging capabilities that may be used while developing CA Enterprise Tester scripts. The CA Enterprise Tester Monitor allows a script to be executed a statement at a time, a session I/O at a time, or to place stops at specified points within a script. Monitor commands support viewing and alteration of script working storage and other capabilities.

The Interactive System Log Facility may be used to capture script logs in a data set. Users may also request a dump of session request unit data for analysis and diagnosis of invalid 3270 commands or other session protocol errors.

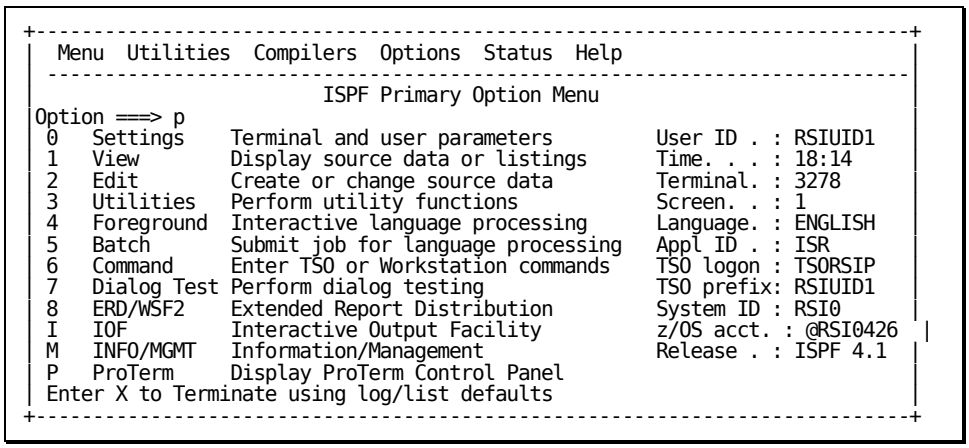
CA Enterprise Tester Utilities are a collection of utility programs providing functions that compliment and otherwise enhance CA Enterprise Tester facilities, including the CA Enterprise Tester Interactive System. CA Enterprise Tester Utilities are documented in the section entitled, See CA Enterprise Tester Utilities.

CA Enterprise Tester provides password security features which permit the capture and use of procedures used to sign on to online systems, and ensure that actual passwords remain confidential. For complete documentation of these features, including Interactive System panels and functions, please refer to Password Security.

This chapter provides an introduction to features of the CA Enterprise Tester Interactive System, including descriptions of Interactive System commands, controls and user options, and describes how these capabilities may be used to support a variety of applications.

CA Enterprise Tester Start-up and Control

The CA Enterprise Tester Interactive System runs under the control of TSO and ISPF. While using ISPF/PDF, CA Enterprise Tester may be accessed using one or more methods that were chosen when CA Enterprise Tester was installed.



CA Enterprise Tester may have been installed as an option on the ISPF/PDF Primary Option Menu or another ISPF selection panel, allowing you to access CA Enterprise Tester by selecting the ProTerm option.

CA Enterprise Tester may have been installed as an ISPF/PDF command, allowing you to access CA Enterprise Tester by entering ProTerm or another command name in the command field of any ISPF/PDF panel.

A CLIST or REXX EXEC may have been installed, allowing you to access CA Enterprise Tester by entering the CLIST or EXEC name at the TSO Command Processor panel (ISPF/PDF option 6) or by entering the TSO command and the CA Enterprise Tester CLIST or EXEC name via the command field of any ISPF panel.

If CA Enterprise Tester is installed as an ISPF/PDF Primary Option Menu option you may select CA Enterprise Tester by entering the menu option selection, P. Alternatively, if CA Enterprise Tester has been installed as an ISPF/PDF command, you may enter the command, ProTerm, into the panel's option field. The method that you use to get to the Control Panel depends on how CA Enterprise Tester was installed at your installation.

When CA Enterprise Tester has been successfully selected, the Control Panel, illustrated in the following example, displays.

Control Panel

The Control Panel is used to start, monitor and control CA Enterprise Tester functions, including sessions, scripts, recordings, traces and utilities.

```
+-------------------------------------------------------------------------------+
|ProTerm Control Panel ------------------------------------ 000 Active Ports|
| COMMAND ===>                                                                  |
|                                                          SCROLL ===> PAGE     |
| S    START/SEL SESSION    R   RECORD A SCRIPT    UT        UTILITY MENU      |
| P    PLAY A SCRIPT        T   TRACE SCREENS      K         KEYS PANEL        |
| A    ANALYZE A SESSION    LOG LOG PANEL          O         OPTIONS PANEL     |
| B    BROWSE TRACE DATA    PF15 END (E P, E 1, ETC)SEC      SECURE DATA       |
|Port     System      Terminal    Session     Script      Record      Trace    |
|----    --------     --------    --------    --------    --------    -------|
| 1                                                                             |
| 2                                                                             |
| 3                                                                             |
| 4                                                                             |
| 5                                                                             |
| 6                                                                             |
| 7                                                                             |
| 8                                                                             |
| 9                                                                             |
| 10                                                                            |
| 11                                                                            |
| 12                                                                            |
+-------------------------------------------------------------------------------+
```

The number of active ports under the control of this Interactive System session is displayed in the upper right corner of the Control Panel. The upper right corner is also used for standard ISPF short message display.

A standard ISPF command entry area on the second line of the panel is available on all CA Enterprise Tester panels, with the exception of full-screen session displays, and permits entry of CA Enterprise Tester commands as well as ISPF commands such as SPLIT and SWAP, and execution of TSO commands.

The third line of the panel is a standard ISPF long message display area. A long message is displayed when you press the PF key to which you have assigned the HELP command, (usually PF1), when a short message is displayed, further describing an error or unusual condition, or giving other additional information. Not all short messages will have additional long messages.

The SCROLL area is a standard ISPF/PDF scroll amount control area. Valid scroll options include PAGE, HALF, and CSR scroll amounts.

Command List

CA Enterprise Tester commands available at the Control Panel are displayed in the active command list. The command list area is dynamic, and will change to reflect the type of functions which are active under the current Interactive System session. CA Enterprise Tester commands available from this panel are described later in this section.

Port List

The body of the Control Panel is a list of ports. A port is a place to anchor a CA Enterprise Tester virtual terminal that is used for a session with an online system.

- A virtual terminal is attached to a port when a SESSION command is entered to start a session.
- Using the PLAY command, a CA Enterprise Tester script may be executed on a port. A script may attach a virtual terminal to a port by executing a Script Language LOGON statement, or a script may use a session that was previously started on the port by a SESSION command.
- A session recording or trace may be started on a port to capture session activity that occurs on the port. While a recording or trace is active, all session events that occur on the port are captured, including transactions entered when a person uses a CA Enterprise Tester session manually and transactions generated by a CA Enterprise Tester script that is played on the port.
- Sessions, scripts, recordings and traces may be started on a single port in any combination and in any order.
- A trace data set browse or trace list browse may be executed on a port to reproduce the previously captured session without actually creating a live session with an online system.
- The CA Enterprise Tester functions that are started on a port operate independently of functions that are started on other ports.

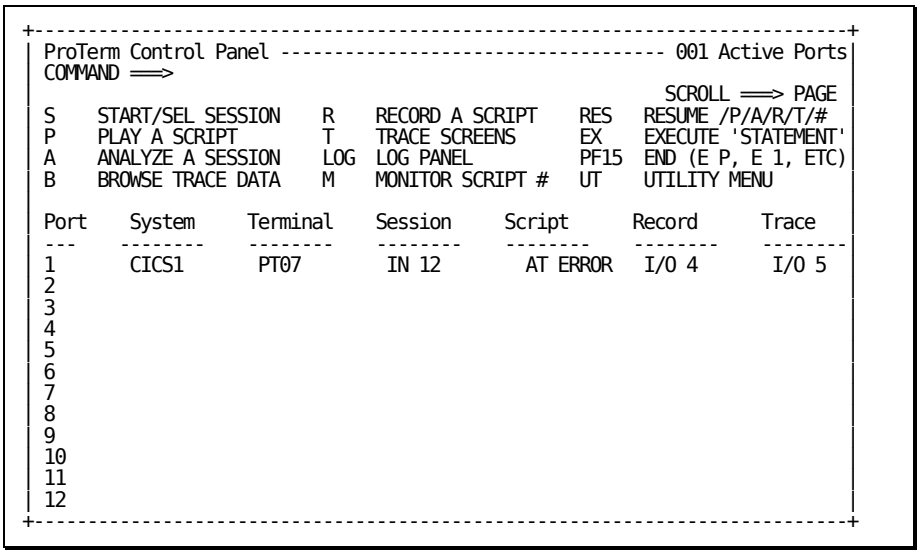
- The Control Panel shows the status of the functions that are active on all displayed CA Enterprise Tester ports.
- The Control Panel port list is scrollable and may be positioned using UP, DOWN and LOCATE commands.

System Name

If a session is active on a port, the VTAM APPLID of the online system currently accessed by the CA Enterprise Tester virtual terminal is listed under the System heading.

Terminal Name

If a session is active on a port, the name of the CA Enterprise Tester virtual terminal is listed under the Terminal heading.



Session Status

If a session is active on a port, the current state of the virtual terminal logical unit is listed under the Session heading. The sum of session inputs and outputs, including entered messages and online system responses, is listed with the current state. Theoretically, the state may be any one of those documented in CA Enterprise Tester Logical Unit States, but it will usually be one of the following:

- IN, means that the state is INBOUND, and the system is waiting on input;
- OUT, means that the state is OUTBOUND, and a terminal is waiting on an online system response;
- I/O, means that the state is CONTENTION, and the session is between transactions;
- CA Enterprise Tester Logical Unit States explains other possible session states.

Script Status

If a CA Enterprise Tester script is active on a port, the Script column will show the status of the script:

- #EX means the number of script statements that have been executed;
- STOPPED means that the script is stopped;
- AT ERROR means that the script is waiting at an error;
- AT REPLY means that the script issued a WTOR and the Interactive System is waiting on a reply from the Interactive System user;
- ENDED means that the script has ended. The ENDED message will be displayed only until the next request has been processed by the Interactive System, and the status columns are refreshed.

Record Status

If a record function is active on a port, the Record column will show the status of the recording:

- I/O shows the sum of script language statements that have been recorded;
- STOPPED means that the recording has been stopped;
- ENDED means that the record function has ended.

Trace Status

If a trace function is active on a port, the Trace column will show the status of the trace activity:

- I/O shows the number of screen images that have been traced;
- STOPPED means that the trace has been stopped;
- ENDED means that the trace function has ended.

Installed Release Identification

The current installed release number is displayed in the lower right corner of the Control Panel.

PF Key Display

You may have a PF key list displayed on the last two or four lines of your panel. If so, this is an ISPF PF key display controlled by the PFSHOW command. You may enter the command, PFSHOW OFF so that you may view the entire panel. You may enable the ISPF PF key display with the PFSHOW ON command, or customize the list with the PFSHOW TAILOR command.

Control Panel Commands

The CA Enterprise Tester Control Panel supports commands that start CA Enterprise Tester functions.

- The SESSION command, abbreviated S, is used to start a CA Enterprise Tester session.
- The PLAY command, abbreviated P, is used to execute a CA Enterprise Tester script.
- The ANALYZE command, abbreviated A, is used to analyze previously traced CA Enterprise Tester sessions.
- The BROWSE command, abbreviated B, is used to browse previously traced CA Enterprise Tester sessions.
- The RECORD command, abbreviated R, is used to start a session recording.
- The TRACE command, abbreviated T, is used to start a session trace.

The CA Enterprise Tester Control Panel supports commands that control CA Enterprise Tester functions.

- The END command, abbreviated E, is used to end CA Enterprise Tester sessions, played scripts, session recordings, browses and traces, an active log, all activity on a port, or to exit CA Enterprise Tester.
- The EXECUTE command, abbreviated EX, is used to enter Script Language statements to alter the speed of or otherwise influence executing CA Enterprise Tester scripts.
- The MONITOR command, abbreviated M, is used to display the CA Enterprise Tester Monitor Panel in order to monitor the execution of a script or to view a session analyzer log.
- The REPLY command, abbreviated REP, is used to respond to a script WTOR statement.
- The RETURN command is used to terminate CA Enterprise Tester functions and exit CA Enterprise Tester. RETURN is implied by the ISPF/PDF JUMP command (=).
- The SESSION command, abbreviated S, is used to select a previously started session for interactive use.
- The STOP command, abbreviated STO, and the RESUME command,, abbreviated RES, stop and resume previously started CA Enterprise Tester scripts, recordings and traces.

The CA Enterprise Tester Control Panel supports commands that provide access to other CA Enterprise Tester panels and functions.

- The KEYS command, abbreviated K, displays the CA Enterprise Tester Keys Panel. The Keys Panel shows current PF key functions and allows you to control the function of all PF keys.
- The LOG command displays the CA Enterprise Tester Log Panel. The Log Panel may be used to open a data set to capture script logs and other session data and messages. If a CA Enterprise Tester Log is currently active, the LOG command will not be listed in the active command list.
- The OPTIONS command, abbreviated O, displays the CA Enterprise Tester Options Panel. The Options Panel is used to view and select CA Enterprise Tester options.
- The SECURE command, abbreviated SEC, displays the security panel for the active password security option. Security panels are used to change passwords represented by tokens, to create password codes, or to select an allowed password security option.
- The TERMINAL command, abbreviated TE, displays the CA Enterprise Tester Terminal Panel. The Terminal Panel may be used to define a default virtual terminal configuration to be used when sessions are started using the SESSION command.
- The UTILITY command, abbreviated UT, displays the CA Enterprise Tester Utility Menu. The Utility Menu may be used to select a CA Enterprise Tester utility option.

Starting Virtual Terminal Sessions

A CA Enterprise Tester virtual terminal session may be started by entering the SESSION command and the name of an online system at the CA Enterprise Tester Control Panel command entry area.

For example:

```
COMMAND ===> s cics1  (and press the Enter key)
```

When you enter the SESSION command, CA Enterprise Tester starts a session between a CA Enterprise Tester virtual terminal and a named online system, in this example, cics1.

The SESSION command supports many optional parameters and may be used to specify logon data to be passed to an online system with a session request.

For example:

```
COMMAND ===> s tso 7 uid1 proc(tsoproc1) 24x80 27x132 extds
```

This form of the command instructs CA Enterprise Tester to:

- Start a session with an online system named: tso
- Use port number: 7
- Pass a user ID to the online system: uid1
- Pass a logon procedure to the online system: proc(tsoproc1)
- The virtual terminal used for the system is configured with 24x80 default screen size, and 27x132 alternate screen size, (a MOD5 3270 terminal);
- The virtual terminal is configured to support the 3270 extended data stream: extds

Complete documentation of all SESSION command parameters may be found in SESSION.

An alternative method of starting sessions using a customized system menu is described in Starting Sessions from a System Menu.

CA Enterprise Tester Session Panel

When a session is started, an online system usually sends a sign on screen to solicit a user ID and password. CA Enterprise Tester waits for and receives any sign-on message and places it in the virtual terminal screen buffer. (If a sign-on message is not received within several seconds, CA Enterprise Tester allows the session to begin with a clear session screen.)

Window Mode/Zoom Mode

Following session initiation and receiving any sign-on message sent by the online system, CA Enterprise Tester displays the virtual terminal screen through a window in the CA Enterprise Tester Session Panel (Window Mode), or uses the full terminal screen (Zoom Mode) to present the session. The initial session presentation is controlled by a user option provided on the CA Enterprise Tester Options Panel.

To contrast the appearance of a CA Enterprise Tester session, when viewed via the CA Enterprise Tester Session Panel in Window Mode, with the appearance of a session presented using the full terminal screen in Zoom Mode, assume that an online system initially solicits a user to sign on by sending a full-screen message that is illustrated in the following example.

```
+--------------------------------------------------------------------------------+
| Hypothetical Online System Logon is now in progress . . .                      |
|                                                                                |
| Enter your user identification, here ==>                                       |
|                                                                                |
|   Enter your user password, here ==>                                           |
|                                                                                |
|   Then press ENTER.                                                            |
+--------------------------------------------------------------------------------+
```

When presented through the CA Enterprise Tester Session Panel Window Mode, the online system message appears as illustrated in the following example.

Note that the first three lines of the Window Mode display are used for standard ISPF title display, command entry area, message area and scroll amount fields, and that the session screen is seen through the window that follows the top three lines.

```
+-------------------------------------------------------------------------------+
|ProTerm Session(PT01/CICS1) ------------------------------------- Port 001 |
| COMMAND ===>                                                  SCROLL ===> CSR |
|----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+---|
| Hypothetical Online System Logon is now in progress . . .                     |
|                                                                               |
| Enter your user identification, here ==>                                      |
|                                                                               |
|   Enter your user password, here ==>                                          |
|                                                                               |
|   Then press ENTER.                                                           |
+-------------------------------------------------------------------------------+
```

ISPF Split-screen Display

The size of the Session window may change. For example, with the cursor positioned on screen row 12, pressing the ISPF SPLIT key (normally PF2 or PF14), splits the screen, and results in the display illustrated in the example below.

```
+-------------------------------------------------------------------------------+
| ProTerm Session(PT01/CICS1) ------------------------------------- Port 001 |
| COMMAND ===>      SCROLL ===> CSR                                             |
|----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+--|
| Hypothetical Online System Logon is now in progress . . .                     |
|                                                                               |
| Enter your user identification, here ==>                                      |
|                                                                               |
|   Enter your user password, here ==>                                          |
|                                                                               |
|   Then press ENTER.                                                           |
|                                                                               |
|  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .|
|    Menu     Utilities     Compilers     Options     Status     Help           |
| ------------------------------------------------------------------------------|
|                        ISPF Primary Option Menu                               |
| Option ===>                                                                   |
|                                                                               |
| 0  Settings     Terminal and user parameters        User ID . : RSIUID6       |
| 1  View         Display source data or listings     Time. . . : 18:14         |
| 2  Edit         Create or change source data        Terminal. : 3278          |
| 3  Utilities    Perform utility functions           Screen. . : 1             |
| 4  Foreground   Interactive language processing     Language. : ENGLISH       |
| 5  Batch        Submit job for language processing  Appl ID . : ISR           |
| 6  Command      Enter TSO or Workstation commands   TSO logon : TSORSIP       |
+-------------------------------------------------------------------------------+
```

While the CA Enterprise Tester session screen may not be fully visible, it is fully functional. In the example, the requested user ID and password may be keyed in where indicated. If the Enter key is pressed while the cursor is in the CA Enterprise Tester portion of the split-screen, then the entered data is placed in the virtual terminal's screen buffer. The modified data from the CA Enterprise Tester screen buffer is then sent to the connected online system as though the virtual terminal keyboard's Enter key had been pressed.

When the online system responds, the response is placed into the virtual terminal's screen buffer. The updated virtual terminal screen is then presented via the Session Panel window.

Scrolling and Switching Display Modes

While viewing a CA Enterprise Tester session through the Session Panel window may be useful in many situations, access to the entire session screen is often a necessity. CA Enterprise Tester provides two ways to gain access to all parts of the virtual terminal screen.

- The scroll commands, UP, DOWN, LEFT and RIGHT, may be used in conjunction with any standard scroll amount, to position the Session Panel window over any portion of the virtual terminal screen.
- The CA Enterprise Tester ZOOM command may be used to switch from ISPF-managed displays, or Window Mode, to TSO full-screen mode, or Zoom Mode, allowing the entire virtual terminal screen to be viewed at once via the real terminal screen.

The Zoom Mode presentation does not include Session Panel control fields, resulting in a display that appears as though the session with the online system were directly between the terminal and the online system.

To re-display a Zoom Mode session display in Window Mode, you may either enter a PF key to which you have assigned the ZOOM command, or press the PA2 key. System actions resulting from the operation of the PA2 key will depend on how you have set the PA2 switch option on page 2 of the Options Panel. CA Enterprise Tester options are described in Options and Controls.

Because ISPF, CA Enterprise Tester, and the online systems that are accessed using CA Enterprise Tester sessions, all provide functions that may be invoked using PF keys, you should become familiar with CA Enterprise Tester PF key customization so that desired functions can be accessed easily using PF keys. CA Enterprise Tester PF key support is described in PF Keys and Commands. Other sections describe how to use some of the special features of CA Enterprise Tester sessions and how to start and use multiple CA Enterprise Tester sessions concurrently.

Starting a Session from a Script

A session with an online system may also be started by playing a CA Enterprise Tester script that starts the session. (Playing Session Scripts describes how to play CA Enterprise Tester scripts.) A script uses the Script Language LOGON statement to start a session. A script may automate other session start-up activities including sign-on and selection of an initial online system application to be presented to the Interactive System user.

A session start-up script may be user-written or may be created by starting a CA Enterprise Tester recording prior to using the SESSION command to initiate a session. (Recording Session Scripts describes how to start a session recording.) While active, the session recording mechanism generates a LOGON statement when a SESSION command is entered and generates TYPE statements when keystrokes are entered to sign on and to invoke other online system functions. The script may be played at a later time to automatically repeat the activity.

PF Keys and Commands

While using the CA Enterprise Tester Interactive System, PF keys have many potential uses:

- ISPF supplied services, including HELP, SPLIT and SWAP, are commonly invoked using PF keys.
- CA Enterprise Tester supports a large number of commands that may be assigned to and may be invoked using PF keys.
- Online systems and applications that are accessed using CA Enterprise Tester sessions often provide functions that are invoked using PF keys.
- CA Enterprise Tester PF key support allows each user to specify the function to be invoked by each of the PF keys. Three sets of PF key assignments may be defined:
- One set to be used when viewing an online session screen display in full-screen, or Zoom Mode;
- A set to be used when using CA Enterprise Tester panels or viewing a session display in Window Mode;
- A third set specifies commands that function as defaults, and are to be used when a Window Mode command is not specified for a PF key or when a specified Window Mode command is not active while using the panel currently displayed.

The CA Enterprise Tester Keys Panel is used to define the function of PF keys while using CA Enterprise Tester. The Keys Panel is displayed when a KEYS command, abbreviated K, is entered in a CA Enterprise Tester panel command entry area, or by a PF key to which the KEYS command has been assigned.

For example:

```
COMMAND ===> k  (and press the Enter key)
```

Keys Panel

The Keys Panel, illustrated in the following example, is displayed when a valid KEYS command is entered from other CA Enterprise Tester panels.

```
+-------------------------------------------------------------------------------+
| ProTerm Keys Panel ----------------------------------------------------------|
| COMMAND ===>                                                                  |
|         ISPF     Window   Zoom              ISPF      Window      Zoom       |
| Key     Command  Command  Command     Key   Command   Command     Command    |
| ----    -------- ------ 1 ------  2   ----  --------  ------ 1    ------  2   |
| PF1  => HELP     PF1      PF1         PF13 => HELP     HELP        KEYS       |
| PF2  => SPLIT    PF2      PF2         PF14 => SPLIT                OPTIONS    |
| PF3  => END      PF3      PF3         PF15 => END      END         END        |
| PF4  => RETURN   PF4      PF4         PF16 => RETURN   CLEAR       CLEAR      |
| PF5  => RFIND    PF5      PF5         PF17 => RFIND    ZOOM        ZOOM       |
| PF6  => RCHANGE  PF6      PF6         PF18 => RCHANGE  ROLL        ROLL       |
| PF7  => UP       PF7      PF7         PF19 => UP       UP          UP         |
| PF8  => DOWN     PF8      PF8         PF20 => DOWN     DOWN        DOWN       |
| PF9  => SWAP     PF9      PF9         PF21 => SWAP                 PA2        |
| PF10 => LEFT     PF10     PF10        PF22 => LEFT     REVIEW      REVIEW     |
| PF11 => RIGHT    PF11     PF11        PF23 => RIGHT    STEP        STEP       |
| PF12 => RETRIEVE PF12     PF12        PF24 => RETRIEVE VIEW        VIEW       |
|Assign ProTerm Commands to Window or Zoom PF keys or enter in COMMAND fields:  |
| Analyze   DOWN    Invite   Monitor   PF1-PF24   RESume   ROll   STEp   UP     |
| ATtn      End     Keys     Options   Play      RETURN   SECur  STOp UTility  |
| Browse    ERase   LEFT  PA1-PA3  Record   REView  Session  TErminal View     |
| Clear     EXecute LOG PEN REPly     RIGHT   SPot Trace    Zoom               |
|1Window Commands override ISPF Commands only if active at the paneldisplayed.  |
|2 Zoom Commands are active while using full-screen Zoom Mode session displays|
+-------------------------------------------------------------------------------+
```

The name of the panel is displayed on the left of the first line of the panel. The upper right corner is used for standard ISPF short message display.

A standard ISPF command entry area on the second line of the panel permits entry of CA Enterprise Tester commands as well as ISPF commands such as SPLIT and SWAP, and execution of TSO commands.

The third line of the panel is the standard ISPF long message display area.

A long message is displayed when you press the PF key to which you have assigned the HELP command, (usually PF1), when a short message is displayed, further describing an error or unusual condition, or giving other additional information. Not all short messages will have additional long messages.

Zoom Command Entry Area

Commands listed in the column labeled Zoom are active only while viewing a CA Enterprise Tester full-screen online system session screen display. ISPF is not active while using a Zoom Mode session display, and therefore only commands directly supported by CA Enterprise Tester may be assigned to Zoom Mode PF keys. Commands directly supported by CA Enterprise Tester are listed at the bottom of the Keys Panel.

Window Command Entry Area

Commands listed in the column labeled Window are active while viewing any CA Enterprise Tester panel, including an online system session screen display in Window Mode. Window commands are CA Enterprise Tester commands that override ISPF commands only while using CA Enterprise Tester panels that support the Window commands. To set up a PF key that is never overridden by a CA Enterprise Tester command, erase the Window command field for the PF key.

ISPF Command Entry Area

Commands listed in the column labeled ISPF are used when a Window Command is not specified for a PF key or when a specified Window Command is not active while using the panel currently displayed.

For example, if the Window Command assigned to PF3 is PF3, and the ISPF Command assigned to PF3 is END, while using a CA Enterprise Tester session in Window Mode, pressing the PF3 key causes a PF3 AID to be sent to the displayed online system session. While using any other CA Enterprise Tester panel, the PF3 key invokes the END command ending the current panel display.

Command List

The commands listed at the bottom of the Keys Panel are directly supported by CA Enterprise Tester and may be entered via the command field of CA Enterprise Tester panels, or may be invoked by pressing a PF key that has been assigned to the command. The listed commands are spelled using dual case letters. The leading capital letters represent an acceptable abbreviation of the command when entered via a CA Enterprise Tester command field. The function of commands are discussed where appropriate, throughout CA Enterprise Tester documentation. Documentation covering all CA Enterprise Tester Interactive System commands is provided in Interactive System Command Reference.

PF Key Display

You may have another PF key list displayed on the last two or four lines of your panel, which may be very confusing. If so, this is an ISPF PF key display controlled by the PFSHOW command. You may enter the command, PFSHOW OFF so that you may view the entire CA Enterprise Tester panel. You may enable the ISPF PF key display with the PFSHOW ON command, or customize the list with the PFSHOW TAILOR command.

Using the Keys Panel

Default ISPF, Window and Zoom Mode command assignments were specified in an Options Table when CA Enterprise Tester was installed in your system, and may differ from the distributed defaults illustrated. You may change your keys to suit your purpose and may change the function of your keys as often as you like.

To assign a command to a PF key, enter the command name into the ISPF, Window or Zoom command entry area that follows the PF key to which the command is to be assigned. The command entry area under each column heading is eight characters long.

Some commands may be entered with additional parameters, for example, STEP 10or REV 10 could be entered under the Window Command column to provide a PF key which would move forward and backward 10 steps at a time within a session review display.

If a command is not available in the mode under which it was entered, CA Enterprise Tester will re-display the previous key assignment when the Enter key is pressed.

The key assignments allow the low-numbered PF keys to be used to access PF key functions provided by an online system, regardless of the display mode being used to view the session. For example, pressing the PF1 key is identical to entering the PF1 command, and the CA Enterprise Tester PF1 command instructs CA Enterprise Tester to send a PF1 key signal on to the online system.

The key assignments illustrated assign high-numbered keys to other CA Enterprise Tester and ISPF supported commands, allowing those commands to be entered by pressing PF keys.

When you use the ZOOM command to switch between a Window Mode and Zoom Mode session display, the function of the PF keys changes for PF keys that are assigned to different commands for Window Mode and Zoom Mode. (You may also use the PA2 key to switch from Zoom Mode to Window Mode, based on an option specified on page 2 of the Options Panel.) If this is confusing, you may simply assign the same commands to both sets of PF keys. However, the ability to assign different commands to each set of keys allows you to solve different problems that are inherent in TSO and ISPF architecture.

Consider the following possible reasons to make your keys behave differently while using sessions in either Window or Zoom Mode:

- While using a session in Zoom Mode, it may be convenient to assign PF key commands to all of the PF keys, so that all PF keys invoke online system services.
- ISPF provides no support for the Clear key. In Zoom Mode, ISPF is not active and the Clear key functions as expected. It is therefore convenient to assign the CLEAR command to a Window Mode key, while this assignment in Zoom Mode may waste a key unnecessarily.
- Specific conventions apply to the PA keys in TSO and ISPF environments. If you are using an online system that requires you to use PA keys, you may assign PA key commands to PF keys, making it easy to send PA key signals to an online system.
- SPLIT, SWAP, HELP and RETRIEVE are frequently used ISPF functions, however, they are directly supported by ISPF and ISPF is not active while using Zoom Mode. This suggests that to access ISPF functions while using Zoom Mode, you must first return to the ISPF-managed Window Mode terminal display. You may do this by either entering the PF key to which you have assigned the ZOOM command, or use the PA2 key to switch from Zoom Mode to Window Mode, based on how you set the PA2 switch option on page 2 of the Options Panel. You may then press a Window Mode PF key to access the desired ISPF command.

Keys Panel Commands

CA Enterprise Tester processing available at the Keys Panel includes the following:

If the Enter key is pressed, or the END command is entered, any changes to the commands assigned to the PF keys are effective immediately and are saved for future Interactive System sessions.

Starting Sessions from a System Menu

CA Enterprise Tester provides facilities which enable the systems programmer who installs CA Enterprise Tester to create a customized list of online systems available in your installation. This procedure is described in CA Enterprise Tester Installation Procedure of the *CA Enterprise Tester System Guide*. Installation procedure options may also limit the systems available for access by CA Enterprise Tester virtual terminals.

CA Enterprise Tester System Menu

The CA Enterprise Tester System Menu is displayed when a SESSION command is entered without an online system name specification.

For example:

```
COMMAND ===> s 3  (and press the Enter key)
```

The example command instructs CA Enterprise Tester to display the CA Enterprise Tester System Menu, and initiate a session on port 003 for the system selected from the System Menu panel.

```
+-------------------------------------------------------------------------------+
| ProTerm System Menu ------------------------------------------- Port 003 |
| COMMAND ===>                                                                  |
|                                                                               |
|     ENTER TO START LOG LOG PANEL    O  OPTIONS PANEL                         |
| PF15 END SESSION REQUEST  K KEYS PANEL   TE  TERMINAL PANEL                   |
|                                                                               |
| To start a session, enter system ID and optional logon data:                  |
|                                                                               |
| Enter system ID ===>                                                          |
|                                                                               |
| Available systems:                                                            |
|                                                                               |
|                                                                               |
| 1 CIT1 - CICS Test Region 1                                                   |
| 2 CIT2 - CICS Test Region 2                                                   |
| 3 IMS1 - IMS Test Region                                                      |
| 4 TSO  - Production TSO                                                       |
| 5 TTSO - Test TSO                                                             |
+-------------------------------------------------------------------------------+
```

The customized menu in your installation may list systems with a short or familiar name, as well as a numeric identifier which may be selected.

The port number to be used for the session is displayed in the upper right corner of the panel. If a specific port number was not entered, CA Enterprise Tester will default to the next available port.

System ID Parameters

You may specify a system identifier from those listed, as well as optional logon data, and when you press the Enter key, the SESSION request will be processed.

Even if an online system is not listed on the customized menu, you may still access the system if you know the VTAM APPLID or other identifiers assigned by the systems programmer. Your ability to access systems not listed on the menu is controlled by system installation options. If you attempt to access what you believe to be a valid system ID, and a logon failed message is displayed, you should contact the systems programmer responsible for CA Enterprise Tester support.

Of course, you must have a valid user ID and password for access to the target system in order to meet your installation security requirements.

PF Key Display

You may have a PF key list displayed on the last two or four lines of your panel. If so, this is an ISPF PF key display controlled by the PFSHOW command. You may enter the command, PFSHOW OFF so that you may view the entire CA Enterprise Tester panel. You may enable the ISPF PF key display with the PFSHOW ON command, or customize the list with the PFSHOW TAILOR command.

System Menu Commands

CA Enterprise Tester commands available at the System Menu are displayed in the active command list. The PF key command assignments may differ from your list based on system installation options and your own PF key customization.

- If the Enter key is pressed, the SESSION command will be processed, initiating a session with the specified online system. The first screen image will be displayed and the virtual terminal keyboard will be free for further input or commands.
- While viewing the System Menu, the END command may be entered to avoid initiating a session, or to return to the Control Panel.
- The LOG command may be used to display the CA Enterprise Tester Log Panel. The Log Panel may be used to control the Interactive System logging function.
- The KEYS command, abbreviated K, may be used to display the CA Enterprise Tester Keys Panel. The Keys Panel may be used to assign commands to PF keys.

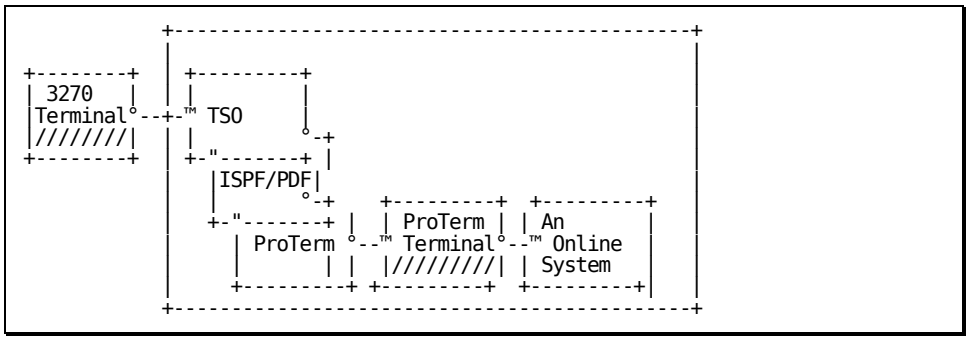
- The OPTIONS command, abbreviated O, may be used to display the CA Enterprise Tester Options Panel. The Options Panel supports a number of options that directly influence the operation of a session. Refer to Options and Controls for a complete description of those options.
- The TERMINAL command, abbreviated TE, may be used to display the CA Enterprise Tester Terminal Options Panel. The Terminal Options Panel may be used to configure the virtual terminal.

Using CA Enterprise Tester Sessions

A CA Enterprise Tester Interactive System session is really two sessions.

- The first session is a session between a real terminal and TSO. The terminal session is used to access TSO, ISPF and CA Enterprise Tester.
- The second session is a session between a CA Enterprise Tester virtual terminal and an online system that is initiated by entering the CA Enterprise Tester SESSION command. CA Enterprise Tester provides the capability to view a virtual terminal screen and to operate a virtual terminal keyboard using a real terminal's screen and keyboard.

The following example illustrates an interactive system session using a virtual terminal with other components in an z/OS system.



Depending on the mode in which a CA Enterprise Tester session is being used, CA Enterprise Tester uses ISPF or TSO screen display services to present virtual terminal screens and to obtain the terminal input that is to be entered into a virtual terminal.

- While displaying a CA Enterprise Tester session through the CA Enterprise Tester Session Panel in Window Mode, CA Enterprise Tester uses ISPF Dialog Management Services to communicate with the real terminal.
- While displaying a CA Enterprise Tester session through a Zoom Mode session display, CA Enterprise Tester uses native TSO display management services, including TPUT and TGET.

Attention Identifier Keys

When an Enter, Clear, PA, or PF key, is pressed while using a CA Enterprise Tester session, the terminal input is first received by TSO or ISPF. Both TSO and ISPF impose some conventions and restrictions on the use of these keys. The following paragraphs describe TSO and ISPF handling of attention identifier keys, (often abbreviated AID keys), and CA Enterprise Tester mechanisms that allow attention keys to be mapped to other keys so that all of the key-activated services provided by systems that are accessed using CA Enterprise Tester sessions, may be accessed using the attention identifier keys on a real terminal keyboard.

Enter Key and Commands

CA Enterprise Tester and ISPF commands may be entered by keying the command into the command field of a CA Enterprise Tester panel and pressing the Enter key. While viewing a CA Enterprise Tester session, if the Enter key is pressed and a command is not entered, then the Enter key is interpreted as a request to press the virtual terminal's Enter key. As a result, CA Enterprise Tester will send a message to the online system that consists of the Enter key's attention identifier, the position of the cursor relative to the virtual terminal screen, and the modified data contained in the virtual terminal screen buffer.

PF Keys and Commands

So that all of the PF key functions supported by all of the systems accessed using a CA Enterprise Tester session may be accessed using PF keys, CA Enterprise Tester provides facilities for assigning commands to PF keys. The list of supported commands provides PF key access to functions supplied by CA Enterprise Tester, by ISPF, and by the online systems that are accessed via CA Enterprise Tester sessions. PF Keys and Commands, describes how to assign commands to PF keys.

Other Attention Identifier (AID) Keys and Commands

CA Enterprise Tester provides commands PF1 through PF24, PA1 through PA3, CLEAR, and ERASE which cause the like-named key on the virtual terminal keyboard to be pressed, and the PEN command, which simulates light pen or cursor select operations.

For example, if the PF1 command is assigned to the PF1 key, when the PF1 key is pressed, a message is sent by CA Enterprise Tester to the connected online system. The message is identical to the message sent on an ordinary terminal session to an online system, when the PF1 key is pressed.

Similarly, if the CLEAR command, is assigned to the PF4 key, then pressing the PF4 key causes the virtual terminal screen to be cleared and causes a message to be sent to the connected online system indicating that the virtual terminal's Clear key has been pressed.

By assigning PFn commands to like-named PF keys, and by using a session in Zoom Mode, CA Enterprise Tester sessions will appear to behave almost exactly like an ordinary terminal session. The only exception to this is the PA2 key which is described below.

PA1 Key

While using a CA Enterprise Tester session, the behavior of the PA1 key depends on the mode in which a CA Enterprise Tester session is being viewed. While viewing a session via the CA Enterprise Tester Window Mode, the function of the PA1 key is fixed by ISPF, and may cause termination of the current ISPF dialog, namely, the CA Enterprise Tester Interactive System.

While using a CA Enterprise Tester session in full-screen Zoom Mode, the PA1 key is sent on to the online system that is connected via the CA Enterprise Tester session. When using applications that issue a terminal read upon receipt of a PA key AID, correct operation of PA key functions may require that the user activate the Read-Modified-All option on the CA Enterprise Tester Terminal Panel.

PA2 Key

The function of the PA2 key is fixed by TSO to be a Reshow key. When the Reshow key is pressed, the active TSO command is required by convention to re-display the last screen sent to the terminal. While a CA Enterprise Tester ISPF panel is displayed, ISPF provides the RESHOW function. The ISPF dialog (CA Enterprise Tester) never sees the PA2 key. When the PA2 key is pressed while viewing a session in full-screen Zoom Mode, CA Enterprise Tester provides the RESHOW function. The CA Enterprise Tester provided RESHOW function, in addition to re-displaying the session screen, may switch to the CA Enterprise Tester Window Mode display mode.

CA Enterprise Tester provides an option on page 2 of the Options Panel which controls the way a PA2 key is processed by the Interactive System. If the PA2 switch option is set to YES, CA Enterprise Tester re-displays a Zoom Mode session display in Window Mode when the PA2 key is pressed. If the option is set to NO, CA Enterprise Tester will stay in Zoom Mode, and entry of the PA2 key will re-display the data previously displayed. To send a PA2 key to an online system, you must enter a PA2 command or assign a PA2 command to a PF key.

Clear Key

The behavior of the Clear key depends on the mode in which a CA Enterprise Tester session is being viewed. While viewing a session in Window Mode, the Clear key causes ISPF screen erasures, which may be corrected by pressing the PA2 key (the Reshow key). While using a CA Enterprise Tester session in full-screen Zoom Mode, the Clear key is sent on to the online system that is connected via the CA Enterprise Tester session.

Light Pen/Cursor Select - Window Mode

While ISPF does not provide general support for pen detectable fields, CA Enterprise Tester provides a PEN command that may be assigned to a Window Mode PF key or may be entered in the Command field while using a session via the CA Enterprise Tester Session Panel. The PEN command simulates light pen or cursor select operations on the field attribute and designator character of the pen detectable field at which the cursor is positioned.

The PEN command is fully documented in PEN.

Zoom Mode Inbound Data Processing

Zoom Mode sessions also support the light pen and cursor select key and provide support for PA1 and PA3 keys.

When using applications that issue a terminal read upon receipt of a pen or PA key AID, correct operation of light pen, cursor select key or PA key functions may require that the user activate the Read-Modified-All option on the CA Enterprise Tester Terminal Panel.

When the user-controlled TSO Read-Modified-All option is active and CA Enterprise Tester subsequently receives a pen or PA AID, a 3270 Read-Modified-All command is sent to the TSO terminal and all modified field data received in the resulting inbound Read-Modified-All data stream is stored in the correct virtual terminal fields even though normal pen or PA key input does not include data keyed into modified fields or does not identify modified screen fields.

In addition, a TSO Read Buffer Option on the CA Enterprise Tester Terminal Panel may be activated to make CA Enterprise Tester Zoom Mode sessions work correctly with IMS/MFS or other applications that issue a 3270 READ BUFFER command upon receipt of 3270 input from an unformatted 3270 screen (a screen that contains no 3270 field attributes).

When a user activates the TSO read-buffer option, and CA Enterprise Tester receives input from an unformatted screen, a 3270 read-buffer command is sent to the TSO terminal and all screen data received in the resulting read-buffer data stream is placed into the virtual terminal screen at the same location it was keyed into the real terminal screen, even though normal inbound data does not include nulls or other data that indicates where data was keyed into a screen.

CA Enterprise Tester Zoom Mode sessions also support applications that use the field-mark or DUP key. (CA Enterprise Tester specifies STFSMODEON,NOEDIT=YES, which turns off TSO interpretation and editing of terminal input.)

ISPF Dialog Restrictions

With the adjustments to attention identifier key usage previously described, most CA Enterprise Tester sessions behave like direct sessions between terminals and online systems. When the Enter key or a PFn, PAn, CLEAR, ERASE or PEN command is entered manually or via a PF key that has been assigned the command, an appropriate message is sent from the virtual terminal to the online system, the online system's response is received and placed in the virtual terminal screen buffer and is then presented at the user's real terminal.

CA Enterprise Tester sessions appear to be a single session rather than two sessions. The mapping of a CA Enterprise Tester session into a TSO terminal session is often completely transparent to a CA Enterprise Tester Interactive System user.

There is, however, one technique used by a few online systems and applications that does not map neatly into an ISPF session. The problem arises only when using a session under ISPF control, and therefore, the easy circumvention of the problem is to use Zoom

Mode for sessions that are subject to the difficulty. The following paragraphs explain the restriction and the CA Enterprise Tester mechanisms that have been provided to compensate for it.

When an ISPF dialog sends a message to a terminal that frees the terminal's keyboard, it is not possible for the dialog to send another message to the terminal until the terminal user returns control to the ISPF dialog by pressing the Enter key or a PF key. While using ISPF terminal display services CA Enterprise Tester is bound by this restriction.

When a CA Enterprise Tester session is being used interactively, the messages that are sent by an online system are received at a virtual terminal and are then sent on to the Interactive System user's real terminal. When CA Enterprise Tester displays the message through a Window Mode Session Panel and frees a user's real terminal keyboard, as directed by a message received from an online system, then CA Enterprise Tester does not get control back from ISPF until the terminal user presses the Enter key or a PF key. If while waiting on terminal input, an online system sends another message to a virtual terminal, the message can not be delivered to the real terminal until terminal input is received.

On an ordinary session between a terminal and an online system, message flow is controlled by the terminal's controller. If an online system frees a terminal keyboard, then at a later time, sends another message to the terminal, then the controller will deliver the message conditionally. If the terminal user has not begun to key data then the message is displayed at the terminal. If the terminal user has begun keying data, then the controller will reject the message. When a message is rejected it is the online system's responsibility to deliver the message at a later time.

While viewing screens in Window Mode, if the terminal keyboard is freed, but an anticipated online system message has not been displayed, it is probable that the message was sent after the online system sent a message that caused the terminal keyboard to be freed.

If the Enter key or a CA Enterprise Tester PF, PA or CLEAR command is entered and a message sent by an online system is waiting to be received, then CA Enterprise Tester will reject the message before sending the message specified by the Enter key or other attention command. This action simulates the action taken by a terminal controller when an online system attempts to send a message after a terminal user has begun to key data.

The CA Enterprise Tester INVITE command may be used to receive messages that have been sent by an online system after the online system has freed the terminal keyboard. If the INVITE command is entered via the Session Panel command field and no options are specified or if the command is entered via a PF key that has been assigned to the INVITE command, then CA Enterprise Tester will immediately check for a message from the online system. If a message is found, it is received and displayed at the user's terminal. If no message is found, the user's keyboard is freed. The INVITE command does not send data to an online system.

By default, CA Enterprise Tester INVITE processing is also performed automatically. After receiving a message from an online system that specifies that a terminal's keyboard is to be freed, CA Enterprise Tester will wait for an automatic INVITE time interval to expire. When the interval has expired, CA

Enterprise Tester will check for another message from the online system. If another message from the online system is found, CA Enterprise Tester will display the previous message but will not free the keyboard. Rather, after displaying the previous message, CA Enterprise Tester will repeat the wait, message check and message display until a message check fails to find another online system message. At that time, the last received message is displayed and the terminal keyboard is freed. Automatic INVITE processing simulates the terminal controller's logic that permits an online system to conditionally deliver messages to a terminal after freeing the terminal keyboard.

The automatic INVITE interval may be controlled via the CA Enterprise Tester Options Panel. The Options Panel is displayed when the OPTIONS command is entered at the CA Enterprise Tester Control Panel or other CA Enterprise Tester panels.

Ending a CA Enterprise Tester Session

A CA Enterprise Tester session may be ended like an ordinary terminal session, by entering the online system's log off request. For example, to end a CA Enterprise Tester session with TSO, enter the LOGOFF command at the TSO READY message. In CICS, enter the CSSF GOODNIGHT transaction or any other transaction that is normally used to terminate a session with CICS.

A CA Enterprise Tester session may also be ended by entering the CA Enterprise Tester END SESSION command. The END SESSION command should be used with caution. If an online application stores information about an active session in an online system data set, and is not given an opportunity to clean up by exiting the application as intended, then use of the END SESSION command may result in stranded online data that may cause unusual application behavior when a subsequent session with the online system accesses the application.

The END command may be entered while using a CA Enterprise Tester session, to exit the display of the session, without terminating the session.

The CA Enterprise Tester Control Panel is displayed when a CA Enterprise Tester session has ended or if the END command is entered to exit the display of a session.

Interactive Session Review

Screen images that represent the most recent messages sent or received via a CA Enterprise Tester Interactive System port, are automatically saved for later review. The saved screen images include those resulting from both interactive and script-driven session activity.

The number of most recent session screens saved is controlled by an option provided on the CA Enterprise Tester Options Panel. The Options Panel is displayed when the OPTIONS command is entered at other CA Enterprise Tester panels.

While viewing a session using the CA Enterprise Tester Session Panel in Window Mode, the Zoom Mode session display, or while using the Monitor Panel, the REVIEW command may be entered to request the display of the previous CA Enterprise Tester session step.

For example, assume that some data was keyed over the last message sent by an online system, as illustrated in the following example.

```
+--------------------------------------------------------------------------+
|ProTerm Session(PT01/CICS) ------------------------------------ Port 001 |
| COMMAND ===>      SCROLL ===> CSR                                        |
|----+----1---+----2---+----3---+----4---+----5----+----6----+----7----+---- |
| this line of data was keyed over the message last sent by cics           |
| STATUS: RESULTS - OVERTYPE TO MODIFY                                     |
| Dat(HFTRFILE) Vsa Clo Ena Rea Upd Add Bro Del Sha   NORMAL               |
|  Obj( CICS.HFTR.FILE     )                                               |
+--------------------------------------------------------------------------+
```

The REVIEW command may be used to display the screen as it appeared the last time CICS sent a message, allowing you to review data that has already been over keyed.

CA Enterprise Tester Session Review Panel

If a REVIEW command is entered at a CA Enterprise Tester virtual terminal session display, either in a Window Mode command data entry area, or with a PF key to which the REVIEW command has been assigned, the CA Enterprise Tester Session Review Panel will be displayed.

```
+--------------------------------------------------------------------------+
| ProTerm Session Review(-1, PT01 <--CD CICS) -------------------- Port 001 |
| COMMAND ===>       SCROLL ===> CSR                                       |
|----+----1----+----2---+---3----+---4----+----5----+----6----+----7----+----|
| S DA(HFTRFILE) ENABLED                                                   |
| STATUS:  RESULTS - OVERTYPE TO MODIFY                                    |
| Dat(HFTRFILE) Vsa Clo Ena Rea Upd Add Bro Del Sha   NORMAL               |
|  Obj( CICS.HFTR.FILE     )                                               |
+--------------------------------------------------------------------------+

+--------------------------------------------------------------------------+
| ProTerm Session Review(-2, PT01 ENTER--> CICS) ----------------- Port 001 |
| COMMAND ===>      SCROLL ===> CSR                                        |
|----+----1---+----2----+---3----+----4---+----5----+----6----+----7----+----|
|cemt s da(hftrfile) enabled                                               |
+--------------------------------------------------------------------------+
```

The sample Session Review Panel displays the following information:

- The review screen title shows the review screen number relative to the current session screen, -1.
- The display reflects the status of the screen following a message that was sent from CICS to the CA Enterprise Tester virtual terminal, PT01. This is indicated by the arrow, <--.
- The CD indicates that the last message received from the online system contained a change direction indicator causing the session direction to be changed to inbound.
- The CD may be replaced by EB, indicating a message that contained an end bracket indicator causing the session to return to the contention state.

Each use of the REVIEW command displays the screen as it appeared following the prior message that was sent to or from the virtual terminal.

Note: that the title line shows the attention identifier key that caused the inbound message, in this case, ENTER.

Session Review Panel Commands

The REVIEW command may optionally specify a number to back up more than one session step at a time. When more than one step is specified and a non-zero step display interval has been specified on the CA Enterprise Tester Options Panel, then each progressively older session step is displayed for the specified time interval before automatically backing up to the prior session step. When the step display interval is zero, intermediate steps are not displayed and the REVIEW command immediately backs up the specified number of steps.

While a review screen is displayed, the STEP command can be used to display the screen that followed the session step currently displayed. The STEP command may also specify a number to advance more than one session step at a time. The presentation of multiple forward steps is also subject to the step display interval. The REVIEW and STEP commands complement each other, allowing you to browse backward and forward in time.

While viewing the most recent review screen, (-1), if a script or session analyzer is not active on the port, the STEP command causes the port's active session to be displayed for interactive use. While viewing any review screen, the SESSION command may be used to present the port's active session for interactive use.

While monitoring a session analyzer or other played script using the Monitor Panel, the REVIEW, SESSION and STEP commands may be entered to display past, present and future virtual terminal screen images. If one of these commands is entered while a script is actively executing, then the script is automatically stopped prior to displaying the requested session screen.

While reviewing session steps on a port on which a script is active, if a STEP command is entered while viewing the most recent session step, then the script is resumed and is executed until the next session step is driven. The active session step is then displayed via the Session Panel.

If, while reviewing a played session, a STEP command is entered that specifies a number of steps greater than the review step currently displayed, then the review is advanced to the most recent review step, decrementing the number entered, the stopped script is resumed and is executed until the specified number of session steps have occurred, after which the current session step is displayed via the Session Panel.

All intermediate steps are displayed for the specified time interval when a non-zero step display time is in effect. The step display time interval is specified on the CA Enterprise Tester Options Panel.

Using Multiple Sessions

When using other CA Enterprise Tester panels, you may enter the END command to end the display of the current panel, and display a higher level panel, eventually displaying the CA Enterprise Tester Control Panel. The Control Panel shows the status of all active functions. The following example illustrates a Control Panel with the status of multiple active sessions listed.

```
+-------------------------------------------------------------------------------+
| ProTerm Control Panel ------------------------------------ 006 Active Ports |
| COMMAND ===>                                                                  |
| S    START/SEL SESSION                                                        |
| P    PLAY A SCRIPT                                                            |
| A    ANALYZE A SESSION                                                        |
| B    BROWSE TRACE DATA                                                        |
|                                                                               |
|   SCROLL ===> PAGE                                                            |
| R RECORD A SCRIPT    RES RESUME /P/A/R/T/#                                    |
| T TRACE SCREENS EX EXECUTE 'STATEMENT'                                        |
| LOG LOG PANEL      PF15 END (E P, E 1, ETC)                                   |
| M MONITOR SCRIPT #   UT UTILITY MENU                                          |
| Port    System      Terminal    Session     Script      Record      Trace    |
|  ---    --------    --------    --------    --------    --------    --------  |
|                                                                               |
| 1        CICSP1       PT07        IN 12       AT          ERROR              |
|                                                                               |
| 2        CICSP2       PT19        IN 36      I/O          4                  |
| 3        CICST1       PT21                   I/O          498     I/O 104    |
| 4        IMS4         PT03        IN 7      STOPPED                           |
| 5        IMS4         PT04        IN 7      STOPPED                           |
| 6        IMS4         PT05        IN 7      STOPPED                           |
| 7                                                                             |
| 8                                                                             |
| 9                                                                             |
| 10                                                                            |
| 11                                                                            |
| 12                                                                            |
+-------------------------------------------------------------------------------+
```

An active session may be selected for interactive use by entering the SESSION command and specifying the session's port number.

For example, to select a session that is active on port 1, enter:

```
COMMAND ===> s 1  (and press the Enter key)
```

The selected session is displayed via the Window Mode Session Panel or in full-screen Zoom Mode, depending on the mode that was last used to view a session. Using the END command to return from a session to the Control Panel ends the display of the session, but does not end the session.

The Interactive System supports a virtual number of concurrent CA Enterprise Tester sessions. To start another session, a SESSION command that specifies the name of the online system with which a session is desired, may be entered at the Control Panel.

For example:

```
COMMAND ===> s cics2  (and press the Enter key)
```

When a new session is started, CA Enterprise Tester connects another virtual terminal to the next unused port (or to a specified port), and uses the terminal to obtain a session with the named online system. Second and subsequent sessions are started and behave in the same manner as the first session.

Note that a CA Enterprise Tester Session Panel title shows both the name of the virtual terminal and the name of the connected online system. Additionally, the title line shows the port number on which a session is active. This information may be used to keep track of which session is currently being used.

When multiple CA Enterprise Tester sessions have been started, switching between sessions may be accomplished in one of two ways.

The END command may be used to return to the CA Enterprise Tester Control Panel, then port number may use the SESSION command to select the desired session.

The CA Enterprise Tester ROLL command may be entered to switch from a session currently being used, directly to an active session on a subsequent port. If the session being displayed is on the highest numbered port that has an active session, then the active session on the lowest numbered port is displayed. By assigning the ROLL command to a PF key, you may switch from session to session by simply pressing the ROLL key.

Recording Session Scripts

Using the CA Enterprise Tester Session Record Facility, CA Enterprise Tester scripts or REXX execs can be created automatically. While a recording is active on a port, CA Enterprise Tester monitors session activity on the port and writes Script Language or REXX statements that reflect the session activity to a specified target data set.

If a recording is active when a session is started on a port, then CA Enterprise Tester records a LOGON statement that specifies the name of the online system with which a session was requested, any logon data that was entered via the SESSION command, and the model of the virtual terminal used for the session.

When a user enters data on a recorded session, CA Enterprise Tester records a TYPE statement that captures the keystrokes entered by the user and the cursor position at which they were entered. Recorded TYPE statements also capture the location of the cursor when the message was sent to an online system and the AID key that caused the message to be sent.

To guard against revealing sign-on passwords to unauthorized persons, the recording mechanism automatically encodes the first word typed into dark modifiable fields. This security feature is fully described in Password Security.

When an online system sends an unsolicited message on a recorded session, or if an online system passes a session to another online system, or stops then restarts session traffic during a session, then an INVITE statement is recorded that allows the recorded script to anticipate and process the session event when the recorded script is executed.

If a recording is active on a port when an END SESSION command is entered to end an active session, then a LOGOFF statement is recorded to end the session when the recorded script is executed.

Assuming that online application software and data is similar to what existed when a script or REXX exec was recorded, the recorded script or exec may be executed to automatically re-create the recorded session events. Recorded session scripts may be executed as a CA Enterprise Tester batch job or may be executed online using the Interactive System PLAY command.

A session recording may be started on an Interactive System port by entering the RECORD command, and an optional port number, at the CA Enterprise Tester Control Panel.

For example:

```
COMMAND ===> r 1  (and press the Enter key)
```

The example specifies that a recording is to be started on Interactive System port number 1. If a port number is not entered, then CA Enterprise Tester will default to the next available port. A RECORD command may also be entered at the Session or Monitor panels, however, because these panels deal with a single port at a time, specification of a port number is neither necessary nor permitted.

CA Enterprise Tester Record Panel

The CA Enterprise Tester Record Panel, illustrated in the following example, is displayed when a valid RECORD command is entered from other CA Enterprise Tester panels.

```
+-------------------------------------------------------------------------------+
| ProTerm Record Panel ------------------------------------------- Port 001    |
| COMMAND ===>                                                                  |
|                                                                               |
|     ENTER TO START                                                            |
| PF15 END RECORD REQUEST                                                       |
|                                                                               |
| Specify target data set or data set and member:                               |
|                                                                               |
| Data Set Name ===> 'UIDRS2.PROTERM.SCRIPT'                                    |
| Member Name   ===> VNDRADD  ( Blank or pattern for member selection list )    |
|   ( Select codes: S Select, B Browse, E Edit )                                |
| Pack data? ===> NO       ( Yes or No )                                        |
|                                                                               |
| Screen comments  ===> 02    ( Record 0 - 43 screen rows before each TYPE )    |
|                                                                               |
| Protect target?  ===> NO       ( Yes or No to replace data )                  |
|                                                                               |
| Share PDS? ===> YES      ( Yes or No )                                        |
|                                                                               |
| Language   ===> PSL      ( PSL or REXX )                                      |
|                                                                               |
| Record data options:    ( Yes or No )                                         |
|                                                                               |
| Delays? ===> YES    MDT fields?   ===> NO                                     |
+-------------------------------------------------------------------------------+
```

The Record Panel is used to initiate a CA Enterprise Tester virtual terminal session recording, and to enter options and specifications which control how the recorded session is to be captured by CA Enterprise Tester.

The name of the panel is displayed on the left of the first line and the number of the port to be used for the recording is displayed on the right. The upper right corner is also used for standard ISPF short message display.

A standard ISPF command entry area on the second line of the panel permits entry of CA Enterprise Tester commands as well as ISPF commands such as SPLIT and SWAP, and execution of TSO commands.

The third line of the panel is the standard ISPF long message display area. A long message is displayed when you press the PF key to which you have assigned the HELP command, (usually PF1), when a short message is displayed, further describing an error or unusual condition, or giving other additional information. Not all short messages will have additional long messages.

Data Set Name Parameters

The name of a target data set for storage of the recorded script may be entered in this area of the Record Panel. The data entry area is a 46 character area, permitting entry of a fully-qualified 44 character data set name enclosed in quotes. You may specify either a sequential data set name or a partitioned data set name, (PDS), and member name where the recorded script is to be saved.

If you do not specify a member name in a PDS, or if you specify a pattern using either the asterisk, * , or the percent sign, % , a directory will be displayed when you press the Enter key, from which a member may be selected. You may also browse or edit members from the directory display. If a command is pending when the directory is displayed, it remains pending, and will be executed when a member is selected.

At entry to the Record Panel, if the data set name area is null, the name is primed according to a system installation option. If the user leaves it there and specifies a member, then the data set is created automatically when the recording is started.

If the message, Dataset not found is displayed in the upper right hand corner of the screen, when you attempt to start a recording, this means CA Enterprise Tester was unable to automatically allocate a script library for you. You may use ISPF's Data Set Utility function, (usually Option 3.2) to allocate the dataset. The only CA Enterprise Tester requirements are that the data set must be allocated with RECFM=FB and LRECL=80.

Pack Data Option

The Record Panel provides an option that may be used to request that scripts be stored using the ISPF packed data format. Packed data may be processed using ISPF utilities such as BROWSE or EDIT, or by CA Enterprise Tester facilities. However, many standard print utilities do not understand the packed data format. Data sets may be packed or unpacked at any time. To pack or unpack a data set, edit the data set using ISPF EDIT, enter the ISPF PACK ON or OFF command, then save the data.

Screen Comments Option

You may specify that lines from the screen image are to be included as comments within the recorded script or exec before recording a TYPE statement, automatically providing documentation. You may enter a number from 0 through 43, to specify the number of lines to include.

Protect Target Option

CA Enterprise Tester provides options to permit you to specify whether or not existing data in either a sequential data set or a member of a PDS should be overlaid with new recorded data. If a protect option of YES is specified, then CA Enterprise Tester will not initiate the record activity which would overlay the existing data, but instead issue a message warning you that the member exists or that the data set is not empty. If a protect option of NO is specified, then the existing data will not be protected and will be replaced by the new recorded script. The initial value for this option, displayed when you enter the panel, may be specified on the second page of the CA Enterprise Tester Options Panel.

Share PDS Option

The PDS share option may be used to allow multiple users to write to the same PDS at the same time. If YES is specified, CA Enterprise Tester dynamically allocates work data sets for the actual recording of script data, and copies the data to the target PDS when the recording is ended.

Directory statistics will be maintained for the PDS based on the options specified on Page 2 of the CA Enterprise Tester Options Panel. For a complete description of CA Enterprise Tester options, refer to Options and Controls.

Language Option

You may specify one of two formats for CA Enterprise Tester to use in recording the session. Option PSL, CA Enterprise Tester Script Language, will record a script which may be executed using the Interactive System Play facility, or from a batch job executing PTEBS. Option REXX specifies that a REXX exec is to be created which may be executed using CA Enterprise Tester's Host Command Environment. (REXX execs will not be stored using the ISPF packed data format).

Record Data Options

CA Enterprise Tester provides user options for whether or not certain types of data should be captured.

- Delays? If YES is specified, CA Enterprise Tester will record DELAY statements, which represent actual user think time in the session. This is the time interval between when the last outbound session screen was transmitted and a subsequent AID key is pressed.

  If NO is specified, DELAY statements will not be recorded. At execution time, a DELAY PERCENT may be specified which can vary the speed at which the script or exec is played.

- MDT Fields? If YES is specified, CA Enterprise Tester will record all fields with the Modified Data Tag on, (MDT). When MDT fields are requested, all input field data is recorded.

If NO is specified, only fields the user has changed are recorded. Changed means that either the MDT was changed or that the data in a field was changed. Rekeying data that is already in a field that was displayed with the MDT initially on does not change the field.

PF Key Display

You may have a PF key list displayed on the last two or four lines of your panel. If so, this is an ISPF PF key display controlled by the PFSHOW command. You may enter the command, PFSHOW OFF so that you may view the entire CA Enterprise Tester panel. You may enable the ISPF PF key display with the PFSHOW ON command, or customize the list with the PFSHOW TAILOR command.

Record Panel Commands

CA Enterprise Tester commands available at the Record Panel include those listed below. The PF key assignment may differ from your own based on system installation options and your customization.

When valid required data has been entered in the Record Panel and the Enter key is pressed, a recording is initiated. After initiating the recording, the panel from which the RECORD command was entered is re-displayed.

While viewing the Record Panel, the END command, or the PF key to which the END command has been assigned may be entered to avoid initiation of a recording.

Additional Record Commands

Commands which control the processing of RECORD requests may be entered from other CA Enterprise Tester panels, including the Control Panel, a Window Mode Session Panel display, or the Monitor Panel.

- After a recording has been started, a STOP RECORD command may be entered to temporarily suspend terminal input monitoring and script statement creation without terminating the recording.
- A RESUME RECORD command may be entered to continue a stopped recording.

- While a recording is active, a RECORD command may be entered at the Session or Monitor panels to add a free-form statement to the recording.

  For example:

  ```
  COMMAND ===> r '* add this comment to active recording'
  ( and press the Enter key )
  ```

  The data to be written must be enclosed in single or double quotes. The data specified is not edited. However, the data specified should be a valid Script Language statement or comment so that the resulting script is executable. The first character of data contained in quotes is written in the first position of the output record in the recorded script, therefore should be a blank, an asterisk indicating the record is a comment, or a script label.

- The END RECORD command may be entered to end a recording. When a recording is ended, the target data set is closed and freed.

Playing Session Scripts

CA Enterprise Tester provides a high-level language, the CA Enterprise Tester Script Language, that may be used to develop executable scripts. CA Enterprise Tester scripts may be user-written or may be created automatically using the CA Enterprise Tester session recording feature as described in Recording Session Scripts. The CA Enterprise Tester Script Language provides complete documentation of the Script Language and describes how to execute CA Enterprise Tester scripts using batch jobs.

This section describes how to execute scripts online, using the CA Enterprise Tester Interactive System. In addition to supporting the execution of CA Enterprise Tester scripts, the Interactive System also supports interactive script debugging, and provides features that may be used to monitor and control scripts while they are executing.

CA Enterprise Tester functions, including sessions, session traces and session recordings may be started on Interactive System ports. CA Enterprise Tester scripts may also be played on CA Enterprise Tester ports. A played script may be used to start a session and to drive the session automatically by entering keystrokes and by examining online system responses.

Interactive System functions active on a port share resources and work together. For example, a session trace or recording may be started on a port to capture activity on a session that is started and used by a person or that is started and used by a script. A person may start a session on a port, then play a script that uses the session to accomplish some desired task. A script may also be played to start a session, sign on for the user, then end, turning the primed session over to a person for interactive use.

A CA Enterprise Tester script may be executed on an Interactive System port by entering the PLAY command, and an optional port number, at the CA Enterprise Tester Control Panel.

For example:

```
COMMAND ===> p 3  (and press the Enter key)
```

The PLAY command example specifies that a CA Enterprise Tester script is to be executed on Interactive System port number 3. If a port number is not specified, CA Enterprise Tester will default to the next available port. The PLAY command may also be entered in the command field of the CA Enterprise Tester Session Panel, however, because the Session Panel deals with a single port at a time, specification of a port number is neither necessary nor permitted.

When the PLAY command is entered, the CA Enterprise Tester Play Panel, illustrated in the following example, is displayed.

CA Enterprise Tester Play Panel

```
+-------------------------------------------------------------------------------+
| ProTerm Play Panel ------------------------------------------- Port 001 |
| COMMAND ===>                                                                  |
|                                                                               |
|     ENTER TO VIEW I/O    M MONITOR EXECUTION  S START SESSION                 |
| PF23 STEP TO SCREEN 1     P PLAY IN BACKGROUND   PF15 END PLAY REQUEST         |
|                                                                               |
| Specify data set containing the script to be played:                          |
|                                                                               |
| Data Set Name ===> 'UIDRS2.PROTERM.SCRIPT'                                    |
| Member Name   ===> VNDRMSTR ( Blank or pattern for member selection list )    |
|   ( Select codes: S Select, B Browse, E Edit )                                |
| Specify optional play options and script parms:                               |
|                                                                               |
| EXEC PARM     ===> SYSID=CICST3                                               |
|   ( ie., LIST,FLOW,variable=value,.. )                                       |
| Specify other script libraries containing linked to scripts:                  |
|                                                                               |
| Library 1     ===> 'UIDRS2.PROTERM.SCRIPT'                                    |
| Library 2     ===> 'APMSTR.PROTERM.SCRIPT'                                    |
| Library 3    ===>                                                             |
|                                                                               |
| Delay Percent ===> 100  ( Delay 0 - 999 percent of DELAY statement times )    |
| Think Time ===> 0000 ( Add think time ssth before all TYPE statements )      |
| Number of Ports ===> 1( 1 - 999  Use ports > 1 for system stress testing )    |
+-------------------------------------------------------------------------------+
```

The Play Panel is used to initiate playing a script on a port, and to enter options and specifications which control how the script is to be processed.

The number of the port on which to play the script is displayed on the right of the panel title line. The upper right corner is also used for standard ISPF short message display.

A standard ISPF command entry area on the second line of the panel permits entry of CA Enterprise Tester commands as well as ISPF commands such as SPLIT and SWAP, and execution of TSO commands.

The third line of the panel is the standard ISPF long message display area. A long message is displayed when you press the PF key to which you have assigned the HELP command, (usually PF1), when a short message is displayed, further describing an error or unusual condition, or giving other additional information. Not all short messages will have additional long messages.

Script Data Set Name

The name of a data set containing the CA Enterprise Tester script to be executed may be entered in this area of the Play Panel. The data entry area is a 46-character area, permitting entry of a fully-qualified 44-character data set name, enclosed in quotes. A user-specified prefix is added to the data set name if the full data set name is not entered in quotes. For a description of prefix options, please refer to Options and Controls.

The data set specified may be either a sequential data set, or a partitioned library with fixed-length, 80-character logical records. If a partitioned library is specified, a member name may also be specified.

If you do not specify a member name in a library, or if you specify a pattern using either the asterisk, * , or the percent sign, % , a library directory or member selection list is displayed when you press the Enter key. Select codes, B, E or S may be used to Browse, Edit or Select a listed member. If a command is pending when the list is displayed, it will remain pending, and executed if a member is selected.

EXEC PARM (Execute Parameters)

The play options and script parameters entered via the Play Panel EXEC PARM field are equivalent to those specified via the JCL EXEC statement PARM field when a CA Enterprise Tester script is executed as a batch job. All system defined parameters are described in Batch JCL. The following list documents some parameters that may be helpful when executing a script from the Interactive System.

Keyword	Description
FLOW	FLOW specifies that each script statement is to be logged as it is about to be executed.
HRESP	HRESP may be used to turn response handling on as though a HANDLE RESPONSES ON statement had been executed prior to executing the script. For a complete description of the Script Language HANDLE RESPONSES statement, refer to RESPONSES.
LIST	LIST specifies that a listing of the script should be written to the log as it is loaded for execution.

You may pass information to a script by specifying keyword variables defined by the script being played.

For example, a script may examine variables named &system and &mask. The following Play Panel EXEC PARM specifications set the variables to specified values prior to script execution:

```
EXEC PARM ===> system=cics1,mask=script1
```

Note: The ampersand, &, is not specified as part of a variable name in the EXEC PARM statement, even though it is a required first character of a variable name within a script.

Any number of keyword options and/or variable field values that can be contained in a 100-character string may be specified. Parameters are separated and delimited by a comma. Spaces may be imbedded within the variable field value.

The EXEC PARM data entry area is a wrap-around area. When you are entering data that exceeds the space available on the first line, the cursor will automatically advance to the next line and you may continue.

Imbedded quotes should be used within a variable field value only when they are required as part of data, such as quotation marks which may be necessary as part of a fully qualified data set name.

For example:

```
EXEC PARM ===> DSN='USER.TEST.DSN(MBR)'
```

Script Libraries

Scripts may use the PSL LINK statement to execute other scripts that are stored as separate members of partitioned script libraries. Play Panel Library fields may be used to specify up to three script libraries that contain other scripts that may be executed using LINK statements.

When the Play Panel is used to play a script, a script named by a LINK statement is located by searching the following libraries:

- Any libraries specified using Play Panel Library fields are searched first and in the order entered.
- If the Play Panel Data Set Name containing the played script is a script library, the library is searched if a linked-to script is not found in any specified Play Panel Library.
- The CA Enterprise Tester system script library is searched if a linked-to script is not found in any library specified using Play Panel fields. The name of the CA Enterprise Tester system script library is specified in CA Enterprise Tester system options and is displayed on the Analyze Panel.

Delay Percent Option

CA Enterprise Tester Script Language DELAY statements may be included when scripts are recorded by the CA Enterprise Tester Record Facility. DELAY statement time intervals represent the amount of time between the last response from an online system, and when the next AID key is pressed by the terminal user. This user think time may be varied by the percent amount entered at this option. An integer value from 1to 999 may be entered.

For example, if 100 is entered, the delay time intervals will be processed at 100 percent, and the script will be played at a speed which closely approximates the speed of the recorded session. If 200 is entered, the script will take twice as long; if 50 is entered it will be twice as fast; if 0 is entered, delay times will be ignored. For more information on the Script Language DELAY statement, refer to the DELAY section.

Think Time Option

You may specify a think time value, in the form ssth, where ss indicates seconds and th indicates tenths and hundredths of seconds, and script execution will be delayed by the think time value each time a transaction is sent to an online system by a Script Language TYPE statement. If DELAY times are included in a recorded script, both delay time values and think time values will affect the speed at which the script is played. For a complete description of the Script Language THINK statement, refer to the THINK section.

Number of Ports Option

This option, when permitted by system options, may be used to play a script on a specified number of ports, up to the displayed maximum value, to create multiple sessions to drive an online system stress test or other multi-session test.

By playing a script that links to different scripts, depending on which port it is executing, any combination of scripts may be played with a single Play Panel request. Multiple Session Testing provides general information and examples that illustrate techniques that may be used to set up and conduct efficient and flexible multi-session online system tests.

PF Key Display

You may have a PF key list displayed on the last two or four lines of your panel. If so, this is an ISPF PF key display controlled by the PFSHOW command. You may enter the command, PFSHOW OFF so that you may view the entire CA Enterprise Tester panel. You may enable the ISPF PF key display with the PFSHOW ON command, or customize the list with the PFSHOW TAILOR command.

Play Panel Commands

CA Enterprise Tester commands available from the Play Panel include the following:

- If the Enter key is pressed to initiate the PLAY function, the default mode for playing a script, the VIEW command, is assumed.
- If the VIEW command is used to initiate script execution, then the script is executed without freeing the terminal keyboard. When script execution has completed, the CA Enterprise Tester panel from which the PLAY command was entered, is re-displayed. A non-zero step display time interval may be specified on the Options Panel to cause the VIEW command to display each online system event created by the script.

- If the STEP command is used to begin script execution, then script execution begins and proceeds until a message is sent or received by the script via a virtual terminal that is attached to the port on which the script is being executed. Script execution is then stopped, and the session step is displayed via the Window Mode Session Panel or Zoom Mode session display.
- If the MONITOR command is used to start a script, then the script is loaded for execution and the CA Enterprise Tester Monitor Panel is displayed. Script execution is deferred until a subsequent command is entered at the Monitor Panel. Script controls and debugging capabilities provided by the Monitor Panel are described in Script Monitoring and Interactive Debug.
- If the PLAY command is used to start script execution, then script execution begins in the TSO background, the CA Enterprise Tester Control Panel is displayed and the terminal keyboard is freed, allowing other terminal activity while the script is executing.
- When entered at the Play Panel, the SESSION command indicates that a script is being played that will start a session for interactive use. When the command is entered, the script is executed. When the script ends, the session is presented for interactive use via the Session Panel in Window Mode or in Zoom Mode, depending on the option specified on the Options Panel for sessions started using the SESSION command. If the script ends and a session is not active, and no other CA Enterprise Tester functions are active, then CA Enterprise Tester returns directly to ISPF.
- While viewing the Record Panel, the END command may be entered to avoid executing a script.
- The OPTIONS command, abbreviated O, may be used to display CA Enterprise Tester Options Panel(s), even though it is not listed on the Play Panel. The Options Panel(s) support options that directly influence how the Interactive System will PLAY a script.
  - When the VIEW command is entered to initiate or resume a PLAY function, a movie is displayed that shows the terminal inputs and online system responses that flow on the session. An Options Panel specification can influence the speed of the movie. The following option setting causes each screen image to be displayed:

    ```
    Step display time for session 'movies'  ===> 0300  (SSTH)
    ```

  - Screen images may be displayed using either Zoom Mode or Window Mode. The Options Panel controls the initial screen image display mode. The following option setting displays screen images using Window Mode:

    ```
    Should PLAY, ANALYZE and BROWSE start in Zoom Mode?  ===> NO (Yes or No)
    ```

    The Window Mode display has the advantage of showing the direction of a message, AID key and other information in the panel title. The Zoom Mode display has the advantage of showing the full session screen.

– A script may contain Script Language THINK and WAIT statements which specify a time interval which will delay script execution. You may specify whether the Interactive System should ignore or honor these time intervals. The following option setting will instruct the Interactive System to ignore the THINK and WAIT statement time intervals:

```
Should THINK and WAIT times delay session 'movies'?  ===> NO (Yes or No)
```

CA Enterprise Tester Session Step Panel

If a CA Enterprise Tester script is played in Window Mode, or Window Mode is invoked with the ZOOM command or key, the CA Enterprise Tester Session Step Panel will be displayed, as illustrated in the following example.

```
+-------------------------------------------------------------------------------+
| ProTerm Session Step(PT02 ENTER--> TSOA118) --------------------- Port 001   |
| COMMAND ===>       SCROLL ===> PAGE                                           |
|----+----1----+----2----+----3---+----4----+---5----+---6----+----7----+----  |
| TSS701I UIDRS2   LAST-USED 17 JUN 08 14:14 SYSTEM=FRM0 FACILITY=TSO           |
| TSS702I COUNT=00049  MODE=FAIL  LOCKTIME=NONE NAME=REED SOFTWARE              |
| UIDRS2 LOGON IN PROGRESS AT 14:16:15 ON JUNE 18, 2008                         |
| NO BROADCAST MESSAGES                                                         |
| EXEC 'UIDRS2.CLIST(SPFP30)'                                                   |
| ***                                                                           |
+-------------------------------------------------------------------------------+
```

In the panel example, the Session Step Panel displays the following information:

- The title of the panel, followed by an open parenthesis, (
- The virtual terminal ID used by the script, PT02
- The attention identifier key which initiated the transaction preceding a directional arrow indicating this is an inbound screen image, ENTER -->
- The name of the online system used by the session followed by a close parenthesis, TSOA118
- The current port on which the script is played, Port 001

Session Step Panel Commands

When a script is stopped at a session step, CA Enterprise Tester commands may be entered in the Window Mode command entry area. CA Enterprise Tester commands available at the Session Step Panel include those listed below.

- The VIEW command, abbreviated V, may be used to resume playing the script, displaying each online system session screen image without freeing the keyboard until script execution has completed.
- The STEP command, abbreviated STE, may be used to advance the script a single session step. An optional number of session steps may be entered.
- The REVIEW command, abbreviated REV, may be used to display a prior session step. An optional number of session steps may be entered.
- The SESSION command, abbreviated S, may be used to display the online system session panel for interactive use through the CA Enterprise Tester Session Panel. By using the SESSION command, you can access the session created or driven by a script. Once you have left an active script and entered data or otherwise manipulated the online system session, you may resume playing the script if you return the session to a point that is logically synchronized with the point at which you entered the SESSION command. You may also use the CA Enterprise Tester Monitor Panel and the EXECUTE command to position the script at a specific point for synchronization.
- The virtual terminal screen image scroll commands, UP, DOWN, RIGHT, and LEFT may be used to view areas of the screen image which may not be displayed in the window you are viewing.
- The END command, abbreviated E, may be entered to return to the CA Enterprise Tester Control Panel. The END command, entered at the Session Step Panel, will not terminate the script. You may end the script by entering the END PLAY command, abbreviated E Pin the Session Step Panel command entry area, or entering the END PLAY port command, abbreviated E P port, from the Control Panel.
- The OPTIONS command, abbreviated O, may be used to display the CA Enterprise Tester Options Panel.
- The KEYS command, abbreviated K, may be used to display the CA Enterprise Tester Keys Panel.

Status of a Played Script

When a CA Enterprise Tester script is executed in the TSO background by entering the PLAY command at the Play Panel, or when a script started using a different command is later released to run in the TSO background by entering a RESUME command, then the status of the script may be viewed via the Control or Monitor Panels.

- The Control Panel shows the status of all active CA Enterprise Tester functions on all ports, including a running count of session messages and script statements executed. While using the Control Panel, the execution of one or all active scripts may be stopped or resumed by entering the STOP PLAY or RESUME PLAY commands. These commands apply to all active or stopped scripts, unless the command specifies a port number. The END PLAY and EXECUTE commands may also be used to effect the execution of one or all executing scripts.
- When the MONITOR command is used to initiate script execution, or when the MONITOR command is entered from the Control or Session Panels, then the detailed status of the script that is executing on a single port is displayed via the Monitor Panel. See

  Script Monitoring and Interactive Debug for details on the Monitor Panel display and the numerous commands and options supported by the CA Enterprise Tester Monitor.
- Once a script is started using a command such as STEP, VIEW or SESSION, if a problem is encountered that is not anticipated by logic within script prior to completion of the script, then script execution is stopped and the script status is displayed via the Monitor Panel.

Free-Form Entry of PLAY Specifications

The Play Panel also supports free-form entry of the data set, member and parameter fields via the panel's command field, and as operands of the PLAY, MONITOR, STEP, VIEW, WAIT, or SESSION commands. This capability, when used in conjunction with ISPF command stacking, allows scripts to be played without the need to display the Play Panel.

For example, while viewing a CA Enterprise Tester session in Window Mode, a script may be executed on the current port by entering the following command:

```
COMMAND ===> p;v proterm.script.lib scriptnm 'parm=value'
```

The command requests the logical display of the Play Panel, then uses the default ISPF command stacking character, semi-colon, ; , to specify data to be entered into unprotected data entry areas on the panel. (The command stacking character or command delimiter is a value specified via ISPF/PDF option 0.1)

The v (VIEW) command specifies that CA Enterprise Tester should display session screen images as the script is executed. The data set name, member name and script parameters following the VIEW command, override previously entered and remembered Play Panel fields. Fields not specified in a free-form command will be picked up from previously entered and saved Play Panel fields.

Script Link Libraries may not be entered as free-form command data.

Stopping a Script

When a CA Enterprise Tester script is executing in the TSO background as a result of entering the PLAY command at the CA Enterprise Tester Play Panel, progress of the script may be monitored by watching the session status data displayed on the CA Enterprise Tester Control Panel. This data will be updated each time the Enter key is pressed.

If you wish to interrupt a script which is running in TSO background, you may enter the STOP PLAY port command, abbreviated STO P port, from the Control Panel. The script will be interrupted when the Script Language statement currently being executed has completed.

You may, however, find a need to interrupt a CA Enterprise Tester script that is executing as you are viewing the resulting session screen displays on your terminal, and your physical terminal keyboard is not free for command input.

- You may wish to interrupt a script movie that is running as a result of a VIEW command;
- Your terminal may be hung waiting on a responses from an online system session with a virtual terminal;
- A script may be looping.

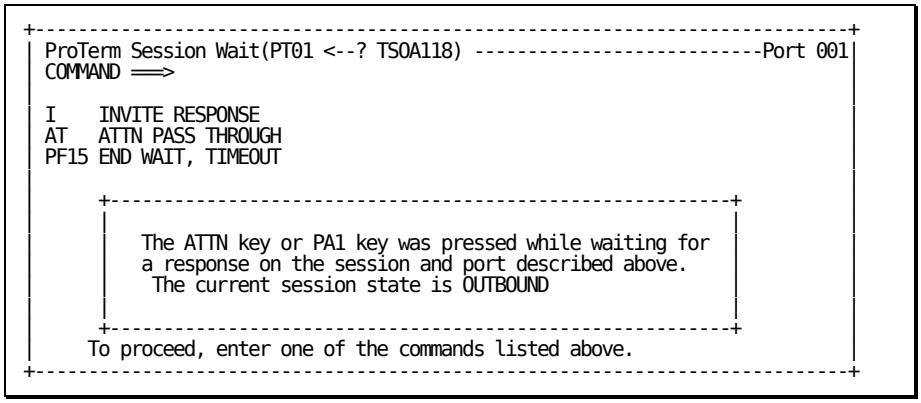
When the physical keyboard is locked, you may press the keyboard Attention key to stop the executing script. CA Enterprise Tester will intercept the Attention key and process the script interruption in different ways, depending on the state of the virtual terminal session with the online system at the microsecond the signal is actually received by CA Enterprise Tester. The actual processing of the Attention key interruption is variable and unpredictable because of the many software and hardware communication components involved between your pressing of the key on your physical terminal and the actual intervention by CA Enterprise Tester in the script processing and the status of the online system session. Data communication transmission times will also affect the Attention key processing. Eventually, CA Enterprise Tester will interrupt the executing script.

CA Enterprise Tester may process the script interruption in one of the following ways.

- CA Enterprise Tester will interrupt script execution and display the CA Enterprise Tester Session Step Panel with the message, ATTN processed in the short message area in the upper right corner of the screen. You may then proceed as you would normally at a script session step. The Session Step Panel is described in CA Enterprise Tester Session Panel.
- CA Enterprise Tester will interrupt script execution and display the CA Enterprise Tester Monitor Panel with the message, ATTN processed in the short message area in the upper right corner of the screen. One of the following messages will be displayed in the Script Status message display area:

  - " SCRIPT IS STOPPED "
  - " SCRIPT IS WAITING AT INBOUND SESSION STEP "
  - " SCRIPT IS WAITING AT OUTBOUND SESSION STEP "

  You may then proceed as you would normally from the Monitor Panel. The Monitor Panel is described in Script Monitoring and Interactive Debug.
- CA Enterprise Tester will interrupt script execution and display the CA Enterprise Tester Session Wait Panel. The Session Wait Panel is illustrated in the following example.



CA Enterprise Tester Session Wait Panel

When a CA Enterprise Tester script is interrupted by an Attention key, the CA Enterprise Tester Session Wait Panel may be displayed, depending on the timing of the processing of the Attention key, and the status of the virtual terminal session.

The Session Wait Panel displays the script interruption description and the command options available for processing.

Session Wait Panel Commands

CA Enterprise Tester commands available at the Session Wait Panel include those listed below:

- The INVITE command, abbreviated I, may be entered to issue a Script Language INVITE to the online system. Once the INVITE statement has been processed, the script will proceed, either resuming the wait on system response, or continuing on to the next session step if an appropriate system response was received.
- The ATTN command, abbreviated AT, may be entered to request that CA Enterprise Tester pass an Attention key on to the online system. The online system will then process the Attention key.
- The END command, abbreviated E, may be entered to issue a Script Language TIMEOUT statement. CA Enterprise Tester will process the TIMEOUT, and display the Monitor Panel with the message, SCRIPT IS WAITING AT ERROR , displayed in the Script Status message display area.
- Further messages will be displayed in the Message Log area describing the status of the online session and processing of the Attention key interruption. These messages may include:

  - `" ATTN INTERRUPT PROCESSED, ACTION IS TIMEOUT ";`
  - `" SESSION TIMED OUT, HANDLE TIMEOUT IS OFF ";`
  - `" STATEMENT AT ERROR: (LINE nn IN SCRIPT 'name') "`

  You may then proceed as you would normally from the Monitor Panel. The Monitor Panel is described in Script Monitoring and Interactive Debug.

Tracing Session Screens

The CA Enterprise Tester screen trace feature is used to automatically capture CA Enterprise Tester virtual terminal screen images when each session input or output event occurs. While a trace is active on a port, CA Enterprise Tester monitors session activity on the port and writes a screen image to a trace data set, each time that a message is sent to or received from an online system. The traced session activity may result from using a CA Enterprise Tester session interactively or may be generated by a CA Enterprise Tester script that is used to drive a session.

Optional trace formats enable the capture of screen images that serve as ideal documentation, or that support the re-creation of the traced session. When a traced session is re-created, live online system responses from the current session, may be compared to traced online system responses from a previous session, to automatically identify any differences in the way the online system has responded.

A CA Enterprise Tester trace may be initiated on an Interactive System port by entering the TRACE command at the CA Enterprise Tester Control Panel.

For example:

```
COMMAND ===> t 1  (and press the Enter key)
```

The example command specifies that a trace is to be started on Interactive System port number 1. If a port number is not entered, CA Enterprise Tester will default to the next available port. A TRACE command may also be entered at the Session or Monitor panels, however, because these panels deal with a single port at a time, specification of a port number is neither necessary nor permitted.

CA Enterprise Tester Trace Panel

The CA Enterprise Tester Trace Panel, illustrated in the following example, is displayed when a valid TRACE command is entered.

```
+------------------------------------------------------------------------------+
| ProTerm Trace Panel ------------------------------------------- Port 001     |
| COMMAND ===>                                                                 |
|                                                                              |
| ENTER TO START                                                               |
| PF15 END TRACE REQUEST                                                       |
|                                                                              |
|Specify target data set, data set and member, or SYSOUT=class,DEST=destname:  |
|                                                                              |
| Data Set Name ===> 'UID1.PROTERM.TRACE'                                      |
| Member Name ===> VNDRADD ( Blank or pattern for member selection list )      |
| ( Select codes: S Select, B Browse, E Edit )                                 |
| Pack data? ===> YES ( Yes or No )                                            |
|                                                                              |
| Initial caption ===>                                                         |
|                                                                              |
| Protect target? ===> NO ( Yes or No to replace data )                        |
|                                                                              |
| Share PDS? ===> YES ( Yes or No )                                            |
|                                                                              |
| Trace type ===> F ( F Format for Analyze, I Image, U Uppercase )             |
|                                                                              |
| Trace data options: ( Yes or No )                                            |
|                                                                              |
| Delays? ===> YES Logons? ===> NO Free KB only? ===> YES                      |
+------------------------------------------------------------------------------+
```

The name of the panel is displayed on the left of the first line and the number of the port to be used for the trace is displayed on the right. The upper right corner is also used for standard ISPF short message display.

A standard ISPF command entry area on the second line of the panel permits entry of CA Enterprise Tester commands as well as ISPF commands such as SPLIT and SWAP, and execution of TSO commands.

The third line of the panel is the standard ISPF long message display area. A long message is displayed when you press the PF key to which you have assigned the HELP command, (usually PF1), when a short message is displayed, further describing an error or unusual condition, or giving other additional information. Not all short messages will have additional long messages.

Data Set Name Parameters

The name of a target data set for storage of traced screen data must be specified in this area of the Trace Panel. The data entry area is a 46-character area, permitting entry of a fully-qualified 44-character data set name, enclosed in quotes. A user-specified prefix is added to the data set name if the full data set name is not entered in quotes. For a description of prefix options, please refer to Options and Controls.

The data set may be:

- A sequential data set with data set characteristics RECFM=FBA and LRECL=135. The 135 character width allows room for a 132 character wide screen, screen borders and an ANSI carriage control character;
- A partitioned data set, (PDS), and member name with data set characteristics RECFM=FBA and LRECL=135; If you do not specify a member name in a PDS, or if you specify a pattern using either the asterisk, * , or the percent sign, %, a library directory or member selection list is displayed when you press the Enter key.

  Select codes, B, E or S may be used to Browse, Editor Select a listed member.

  If a command is pending when the list is displayed, it will remain pending, and executed if a member is selected.
- A JES SYSOUT class and optional destination name. When a trace is written to a SYSOUT data set, IMAGE and NOPACK options are forced.

Following ISPF conventions, CA Enterprise Tester will save many displayed panel variables when a user completes a task. The first time a user displays the CA Enterprise Tester Trace Panel, there will be no saved data set name. When a data set name area is null, CA Enterprise Tester will prime the name according to a data set naming convention selected as a system installation option. If the user leaves the name unchanged and specifies a member, then the data set, a PDS, is created automatically when the trace is started. You may use ISPF's Data Set Utility function, (usually Option 3.2) to allocate a trace data set. The only CA Enterprise Tester requirements are that the data set must be allocated with RECFM=FBA and LRECL=135.

Pack Data Option

The Trace Panel provides an option that may be used to request that traced screens be stored using ISPF packed data format. A packed trace typically requires less than 20% of the disk space required for a trace that is not packed. Packed data may be processed using ISPF utilities such as Browse, View or Edit, and by all CA Enterprise Tester facilities. However, many standard print utilities do not understand the packed data format.

Data sets may be packed or unpacked at any time. To pack or unpack a data set, edit the data set using ISPF Edit, then enter the PACK ON or PACK OFF command, which will update the ISPF Edit Profile, then save the data. All ISPF Profile parameters may be examined by entering the PROFILE command when using an ISPF Edit work space.

Initial Caption Data Entry Area

Captions may be placed on traced screen images. The Trace Panel provides a field to specify an initial screen caption. While a trace is active, the caption may be changed by entering a TRACE command that specifies a new caption in the Session Panel command field.

For example:

```
COMMAND ===>  'this is a new caption'
```

Protect Target Option

CA Enterprise Tester provides options to permit you to specify whether or not existing data in either a sequential data set or a member of a PDS should be overlaid with new traced data. If a protect option of YES is specified, then CA Enterprise Tester will not initiate the trace activity which would overlay the existing data, but instead issue a message warning you that the member exists or that the data set is not empty. If a protect option of NO is specified, then the existing data will not be protected and will be replaced by the new traced data. The initial value for this option, displayed when you enter the panel, may be specified on the second page of the CA Enterprise Tester Options Panel.

Share PDS Option

The PDS share option may be used to allow multiple users to write to the same PDS at the same time. If YES is specified, CA Enterprise Tester will provide dynamically allocated work data sets for the actual recording of trace data, and copy the data to the target PDS when the trace is ended.

Directory statistics will be maintained for the PDS based on the options specified on Page 2 of the CA Enterprise Tester Options Panel. For a complete description of CA Enterprise Tester options, refer to Options and Controls.

Trace Type Options

You may specify the type of trace to be captured:

- F, specifies a CA Enterprise Tester FORMAT trace is to be initiated. Screens captured using the format option contain coded or actual screen field attributes as documented in 3270 Attribute Values. When the 3270 extended data stream is used for a session, the extended attribute buffer is also written to the trace data set.

  Format trace data sets may be printed or converted to other formats, or may be converted to CA Enterprise Tester scripts or REXX using CA Enterprise Tester Trace or Record Utilities. Refer to CA Enterprise Tester Utilities for details.

- I, specifies a CA Enterprise Tester IMAGE trace is to be initiated. When an image trace is specified, screens written to the trace data set are edited. Screen attributes, non-displayable characters including nulls, and screen data contained in non-display screen fields, are replaced with blanks. Graphic escape characters are replaced with dashes, (-). The resulting screen image looks like an actual terminal screen image would look. The screen images may be used to document session events or may be copied from the trace data set into other online system or application documentation and training material.

- U, specifies a CA Enterprise Tester UPPERCASE IMAGE trace is to be initiated. In addition to the editing performed when the image trace option is specified, when the uppercase format is specified, all lowercase alphabetic characters are converted to uppercase, including screen and page titles. The uppercase option creates a trace which may be printed on a printer that does not support lowercase data.

Trace Data Options

CA Enterprise Tester provides user options for whether or not certain types of data should be captured.

- Delays? If YES is specified, PSL DELAY statements are written to the trace dataset that capture the actual user think that preceded each traced inbound screen image. CA Enterprise Tester session analyzers may execute traced DELAY statements, to improve session synchronization when sessions with IMS or other systems that may respond differently when transactions are repeated, depending on current system load or other factors.

  If NO is specified, DELAY statements will not be included in the trace.

- Logons? If YES is specified, a PSL LOGON statement is written to the trace dataset when a session is started using the SESSION command, that includes any logon data or other information specified by the command. PSL statements captured in traced session data are returned by the PSL TRACK statement and may be executed by CA Enterprise Tester session analyzers to start sessions that are identical to an original session started using a SESSION command.
- Free KB Only? If YES is specified, only outbound screens that free the keyboard are to be traced. The space required to store a trace is reduced when only Free KB screens are traced.

PF Key Display

You may have a PF key list displayed on the last two or four lines of your panel. If so, this is an ISPF PF key display controlled by the PFSHOW command. You may enter the command, PFSHOW OFF so that you may view the entire CA Enterprise Tester panel. You may enable the ISPF PF key display with the PFSHOW ON command, or customize the list with the PFSHOW TAILOR command.

Trace Panel Commands

CA Enterprise Tester commands available for execution at the Trace Panel include those listed below. The PF key assignment may differ from your own based on system installation options and your customization.

When valid required data has been entered in the Trace Panel and the Enter key is pressed, a trace is initiated. After initiating the trace, the panel from which the TRACE command was entered is re-displayed.

While viewing the Trace Panel, the END command, or the PF key to which the END command has been assigned may be entered to avoid initiation of a trace.

Additional Trace Commands

The following commands may be entered from other CA Enterprise Tester panels, such as the Control Panel, the Monitor Panel, or a Window Mode session display command entry area:

After a trace has been started, a STOP TRACE command may be entered to temporarily suspend screen image tracing without closing the trace data set.

You can enter a RESUME TRACE command to continue a stopped trace.

You can enter the END TRACE command may be entered to terminate a trace. When a trace is ended, the target data set is closed and freed.

Sample CA Enterprise Tester Traces

While a trace is active on a port, screen images are captured that reflect all virtual terminal inputs and responses that occur on a session that is on the same port, including session I/O resulting from manual use of a session and session I/O that is driven by a CA Enterprise Tester script that is played on the port.

The screen images in examples which follow illustrate CA Enterprise Tester trace output. The screen images were taken from CA Enterprise Tester trace data sets created during a virtual terminal session with TSO and ISPF/PDF.

Characteristics describing the online system session and the virtual terminal are described in values documented in the trace header record which precedes session screen images.

Characteristics describing each inbound or outbound step in the session are described in values documented in the description line which precedes the actual screen image data.

In the first screen image of the image trace example, the following information may be found in the trace header:

- The number one in the first position identifies the trace header;
- The trace header title documents the type of trace: ProTerm Screen Image Trace
- The remaining data is justified to the right margin in the 135-character trace data set, but is moved to the left in the sample;
- The Primary Logical Unit, or name of the online system: PLU TSO
- The Secondary Logical Unit, or name of the virtual terminal: SLU PT01
- Virtual terminal characteristics are documented, the terminal was a standard 24 row by 80 column primary and alternate screen size, and the extended data stream feature was supported: MODEL 24X80 EXTDS
- The date the trace was captured is documented: 01/16/08

Following the trace header record, and before the actual screen image, a CA Enterprise Tester Script Language DELAY statement was captured, documenting the amount of time between the last outbound step from the online system; and when the user pressed an attention identifier or AID key.

In this example, the time interval was nearly 50 seconds: DELAY 4998

The next line is a screen image border record giving information about an inbound session step:

- The name of the virtual terminal in use: PT01
- The Enter key is documented as the attention identifier key pressed, along with an arrow, which indicates the screen image was inbound from a virtual terminal ID to an application ID: ENTER-->
- The message was sent to a TSO system. The actual system identifier was assigned by TSO during session initiation: TSOA028
- The cursor position is documented at row 4, column 15: Cursor(4,15)
- A trace data set screen image sequence number, along with a time stamp is next:

  ```
  #00001 11:50:02.94
  ```

ISPF responded to the request to display the View Entry Panel and an outbound screen image was captured in the trace data set.

The image trace session step description now lists the following information:

- The direction arrow now indicates an outbound screen image and that the message was sent from TSO to the virtual terminal: PT01 <--CD TSOA028
- The CD indicates that the message contained a change direction indicator to change session direction.
- The new cursor position is given: Cursor(10,19)
- And the next screen image sequence number, along with a time stamp is next:

  ```
  #00002 11:50:03.11
  ```

Other information may be contained in a trace heading that indicates protocols and techniques used by an online system. For example, some systems use bracket protocol rather than change direction protocol. The protocol used may be identified by indicators contained in a trace heading.

The <--EB in the following example indicates that the message sent by an online system freed the terminal by sending an end bracket indicator. End bracket additionally places a session in a contention state.

For example:

```
" . . . PT01 <--EB CICST1 . . . "
```

Some online systems send multiple consecutive outbound messages to a terminal to construct a single screen image. The trace mechanism captures only those screen images that are presented to the person or script that is using the session. All session messages are presented to a person when a CA Enterprise Tester session is used interactively. Normally, only session messages that contain a Change Direction, (CD), or End Bracket, (EB), signal are presented to a CA Enterprise Tester script. The result is that some traces may contain multiple outbound screen images while other traces contain only screen images that free the terminal keyboard by sending CD or EB indicators.

If a single traced screen image represents the receipt of multiple consecutive outbound messages that were used to construct a single screen image, then the number of messages that contributed to the traced screen image is documented in the trace heading, immediately following the message direction arrow and preceding the CD or EB indicator.

The following example indicates that 4 consecutive messages were sent by an online system before the terminal keyboard was freed.

For example:

```
" . . . PT01 <--4CD ATSO1002 . . . "
```

The Free KB Only? option on the CA Enterprise Tester Trace Panel is provided so that you may instruct CA Enterprise Tester to capture only outbound messages which free the keyboard, or to capture each individual transmission as they are sent to the virtual terminal.

If you compare the two trace examples, you will see that the FORMAT trace contains the same documentation of session and terminal information, and that an additional buffer of information was captured for each screen image. This buffer contains data required to support the 3270 extended data stream. Also you will note characters representing attributes and other binary data in the screen images.

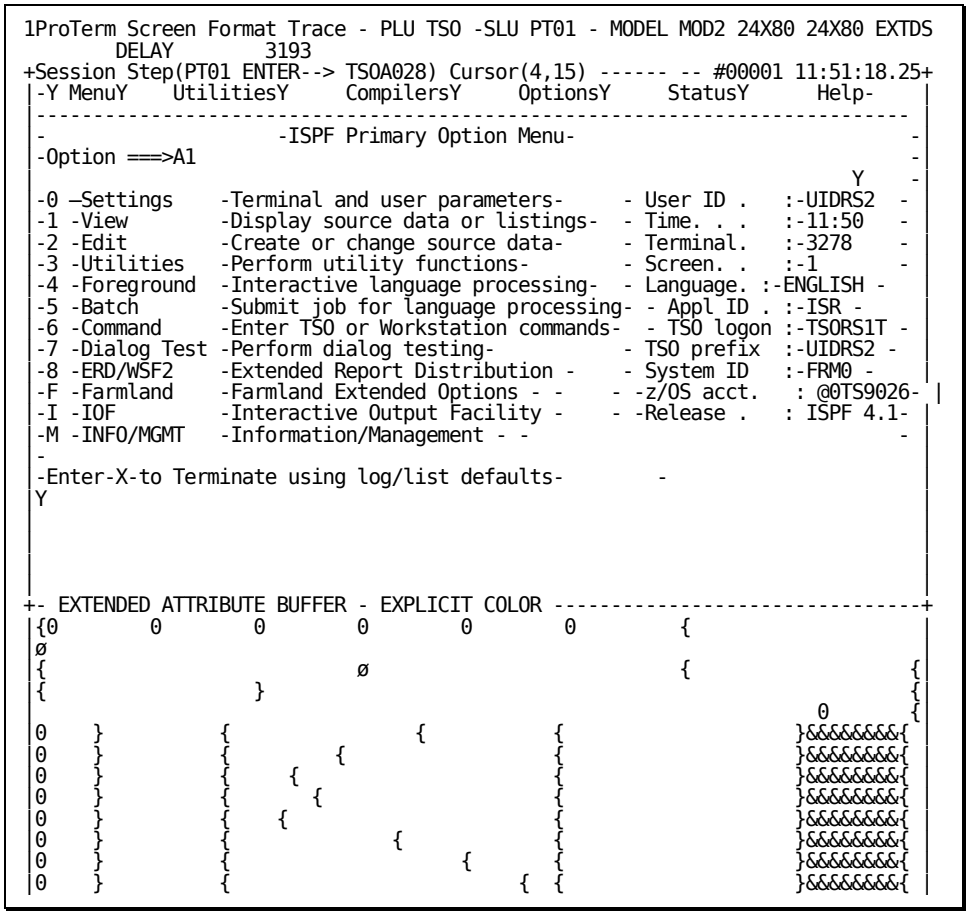
The following example illustrates the IMAGE option part one:

```
1ProTerm Screen Image Trace - PLU TSO - SLU PT01 - MODEL MOD2 24X80 24X80 EXTDS
         DELAY                      4998
+Session Step(PT01 ENTER--> TSOA028) Cursor(4,15) --------- #00001 11:50:02.94+
| Menu       Utilities       Compilers       Options       Status       Help   |
| ------------------------------------------------------------------------------ |
|                          ISPF Primary Option Menu                              |
| Option ===> 1                                                                  |
|                                                                                |
| 0     Settings Terminal and user parameters       User ID . : UIDRS1           |
| 1     View Display source data or listings        Time. . . : 11:49            |
| 2     Edit Create or change source data           Terminal. : 3278             |
| 3     Utilities Perform utility functions         Screen. . : 1                |
| 4     Foreground Interactive language processing  Language. : ENGLISH          |
| 5     Batch Submit job for language processing    Appl ID . : ISR              |
| 6     Command Enter TSO or Workstation commands   TSO logon : TSORS1T          |
| 7     Dialog Test Perform dialog testing          TSO prefix: UIDRS1           |
| 8     ERD/WSF2 Extended Report Distribution       System ID : FRM0             |
| I     IOF Interactive Output Facility             z/OS acct. : @RSI0426       |
| M     INFO/MGMT Information/Management            Release . : ISPF 4.1         |
| P     ProTerm Display ProTerm Control Panel                                    |
|                                                                                |
| Enter X to Terminate using log/list defaults                                   |
|                                                                                |
|                                                                                |
|                                                                                |
|                                                                                |
+--------------------------------------------------------------------------------+
0+Session Step(PT01 <--CD TSOA028) Cursor(10,19) ---------- #00002 11:50:03.11+
| Menu         RefList          RefMode          Utilities          Help       |
| ------------------------------------------------------------------------------ |
| View Entry Panel                                                               |
| Command ===>                                                                   |
|                                                                                |
| ISPF Library:                                                                  |
|    Project . . . RS1                                                           |
|    Group . . . . PROTERM . . .           . . .           . . .                 |
|    Type . . . . SCRIPT                                                         |
|    Member . . . (Blank or pattern for member selection list)                   |
|                                                                                |
| Other Partitioned or Sequential Data Set:                                      |
|    Data Set Name . . .                                                         |
|    Volume Serial . . .       (If not cataloged)                                |
|                                                                                |
| Initial Macro . . . .                Enter "/" to select option                |
| Profile Name . . . . .               /    Browse Mode                          |
| Format Name . . . . .                Confirm Cancel/Move/Replace               |
|                                      Mixed Mode                                |
|                                                                                |
| Data Set Password . .                (If password protected)                   |
|                                                                                |
|                                                                                |
|                                                                                |
+--------------------------------------------------------------------------------+
```

The following example illustrates the IMAGE option part two:

```
1ProTerm Screen Image Trace - PLU TSO - SLU PT01 - MODEL MOD2 24X80 24X80 EXTDS
           DELAY          0551
+Session Step(PT01 ENTER--> TSOA028) Cursor(10,23) -------- #00003 11:50:08.62+
| Menu       RefList       RefMode       Utilities       Help                 |
| ------------------------------------------------------------------------------ |
|                          View Entry Panel                                    |
| Command ===>                                                                 |
|                                                                              |
| ISPF Library:                                                                |
|    Project . . . RS1                                                         |
|    Group . . . . PROTERM . . .      . . .      . . .                         |
|    Type . . . . SCRIPT                                                       |
|    Member . . . pte* (Blank or pattern for member selection list)            |
|                                                                              |
| Other Partitioned or Sequential Data Set:                                    |
|    Data Set Name . . .                                                       |
|    Volume Serial . . . (If not cataloged)                                    |
|                                                                              |
| Initial Macro . . . .              Enter "/" to select option                |
| Profile Name . . . . .             / Browse Mode                             |
| Format Name . . . . .              Confirm Cancel/Move/Replace               |
|                                    Mixed Mode                                |
|                                                                              |
| Data Set Password . .             (If password protected)                    |
|                                                                              |
|                                                                              |
|                                                                              |
+------------------------------------------------------------------------------+
0+Session Step(PT01 <--CD TSOA028) Cursor(4,15) ----------- #00004 11:50:08.98+
| Menu    Functions    Utilities    Help                                       |
| ------------------------------------------------------------------------------ |
| BROWSE      RS1.PROTERM.SCRIPT                          Row 00001 of 00045 |
| Command ===>                                              Scroll ===> PAGE |
|    Name          VV MM  Created   Changed         Size Init   Mod    ID      |
| . PTEAFPDS       01.02  06/06/13  06/06/13 13:16    36 21       0   UIDRS1 |
| . PTEBOOK        01.06  06/07/13  05/06/10 09:31    66 59       0   UIDRS1 |
| . PTECDATE       01.01  05/12/09  05/12/14 12:21   278 278      0   UIDRS1 |
| . PTECLOS1       01.09  04/09/29  08/01/12 13:24    61 41       0   UIDRS1 |
| . PTECLOS2       01.01  04/09/29  04/10/03 06:28    58 58       0   UIDRS1 |
| . PTECODE        01.01  06/01/03  06/01/03 14:00    51 51       0   UIDRS1 |
| . PTECODEU       01.01  06/01/03  06/01/03 14:01    53 53       0   UIDRS1 |
| . PTESA          01.30  08/11/15  08/12/01 15:00  1573 064      0   UIDRS1 |
| . PTESAFE        01.02  08/11/30  08/12/04 15:55    20 20       0   UIDRS1 |
| . PTESAVE        01.01  08/11/31  08/12/04 20:19    20 20       0   UIDRS1 |
:                                                                              :
:                                                                              :
:                                                                              :
```

The following example illustrates the FORMAT option part one:



The following example illustrates the FORMAT option part two:

```
+-------------------------------------------------------------------------------+
1ProTerm Screen Format Trace - PLU TSO -SLU PT01 - MODEL MOD2 24X80 24X80 EXTDS
+Session Step(PT01 <--CD TSOA028) Cursor(10,19) ----------- #00002 11:51:18.28+
|-Y MenuY      RefListY      RefModeY       UtilitiesY      Help-              |
|-------------------------------------------------------------------------------|
|-                        -View Entry Panel-                                   -
|-Command ===>                                                                 -
|                                                                          Y    -
|YISPF Library: Y                                                              -
|- --Project . . . RS1      -                                                   |
|- --Group . . . . PROTERM -. . .    -. . .     -. . . -                       |
|- --Type . . . . SCRIPT       -                                               |
|- --Member . . .     -     -(Blank or pattern for member selection list)-      |
|-                                                                             |
|YOther Partitioned or Sequential Data Set: Y                                  -
|- --Data Set Name . . .                                                       |
|- --Volume Serial . . . - -(If not cataloged)-                                |
|00                                                                            |
|-Initial Macro . . . . -0 --Enter "/" to select option -                      |
|-Profile Name . . . . . - /0-Browse Mode- -                                   |
|-Format Name . . . . . - 0-Confirm Cancel/Move/Replace- -                     |
| 0-Mixed Mode- -                                                              |
|00                                                                            |
|-Data Set Password . .< --(If password protected) -                           |
|                                                                              |
|                                                                              |
|Y                                                                             |
+- EXTENDED ATTRIBUTE BUFFER - EXPLICIT COLOR -------------------------------+
```

Browsing a Traced Session

The Interactive System Browse Facility provides features that may be used to automatically create a movie of a previous session with an online system which was captured using the CA Enterprise Tester Trace Facility. The session will be reproduced using the screen images from the trace list or trace data set without actually creating a live online system session. Session events may be reproduced for demonstration purposes without affecting the operation of the online system or the data accessed by the system. The process of capturing a trace data set is documented in Tracing Session Screens. The trace list feature is documented in Session Analyzer Options.

A browse may be displayed as an uninterrupted movie, with each screen image displayed on the terminal for the amount of time specified in the CA Enterprise Tester Options Panel. You may also display specific screen images or proceed through the images in one or more session steps.

When a browse is initiated, CA Enterprise Tester will number each screen image in the simulated session. You may then move back and forward in time within the browse to a specific screen image by using the LOCATE command.

You may also use the STEP and REVIEW commands to move back and forward in time relative to the currently displayed screen image.

Initiating a Browse

A CA Enterprise Tester trace browse may be initiated on an Interactive System port by entering the BROWSE command at the CA Enterprise Tester Control Panel.

For example:

```
COMMAND ===> b 1 (and press the Enter key)
```

The example command specifies that a browse is to be started on Interactive System port number 1. If a port number is not entered, CA Enterprise Tester will default to the next available port.

The CA Enterprise Tester Browse Panel displays when a valid BROWSE command is entered.

CA Enterprise Tester Browse Panel

```
+-------------------------------------------------------------------------------+
| ProTerm Browse Panel ------------------------------------------- Port 001 |
| COMMAND ===>                                                                  |
|     ENTER TO VIEW I/O         L        LOCATE SCREEN #      K KEYS PANEL    |
|     PF23 STEP TO SCREEN 1     PF15     END BROWSE REQUEST   O OPTIONS PANEL|
| Specify trace data set or trace list data set to be browsed:                  |
|                                                                               |
|    Data Set Name ===>                                                         |
|    Member Name  ===>   ( Blank or pattern for member selection list )         |
|                        ( Select codes: S Select, B Browse, E Edit )           |
| Specify response screens to be displayed (input is always displayed):         |
|                                                                               |
|    Responses     ===> A ( A All responses )                                   |
|                         ( F all responses that Free keyboard )                |
|                         ( T final Transaction responses only )                |
|                                                                               |
| If your device and data support color, specify Zoom Mode attributes:          |
|                                                                               |
| Attributes    ===> E    ( E Extended attributes, B Basic )                    |
+-------------------------------------------------------------------------------+
```

The CA Enterprise Tester Browse Panel is used to initiate a trace data set or trace list browse on a port, and to enter options and specifications which control how the browse is to be processed.

The name of the panel is displayed on the left of the first line and the number of the port on which to initiate the browse is displayed on the right. The upper right corner is also used for standard ISPF short message display.

A standard ISPF command entry area on the second line of the panel permits entry of CA Enterprise Tester commands as well as ISPF commands such as SPLIT and SWAP, and execution of TSO commands.

The third line of the panel is the standard ISPF long message display area. A long message is displayed when you press the PF key to which you have assigned the HELP command, (usually PF1), when a short message is displayed, further describing an error or unusual condition, or giving other additional information. Not all short messages will have additional long messages.

Data Set Name Parameters

The name of a data set containing the CA Enterprise Tester trace or trace list may be entered in this area of the Browse Panel. The data entry area is a 46 character area, permitting entry of a fully-qualified 44 character data set name enclosed in quotes. You may specify either a data set or a member of a partitioned data set, (PDS), which contains

a single session trace, or you may specify a trace list which contains a number of session traces which will be concatenated and displayed in the order in which they are listed to create what appears to be a single session.

Trace lists which were created as input to a CA Enterprise Tester session analyzer may be used for a browse. Keywords such as MASK or SYNC which apply to analyzer scripts will be ignored by browse processing.

If a PDS is specified, a member name may also be entered. If you do not specify a member name in a PDS, or if you specify a pattern using either the asterisk, * , or the percent sign, % , a directory will be displayed when you press the Enter key, from which a member may be selected. You may also browse (using the ISPF/PDF browse function), or edit members from the directory display. If a command is pending when the directory is displayed, it will remain pending, and be executed after a member is selected.

Responses Option

The Browse Panel responses option allows you to specify the type of outbound screen image responses from the online system you want displayed. A session step is a screen image representing a terminal input or online system response. Although a user may enter any number of keystrokes, including Tab keys and Erase EOF keys, all data is combined to form a single input screen image to represent the user view of what has been transmitted to the online system. Online systems, however, may transmit more than one system response forming the user view of the final output screen image. All terminal input or inbound screen images are displayed. You may specify one of the following forms of system response or outbound screen images for display:

A, indicating all outbound system responses should be displayed. This may result in multiple outbound screen images as the result of a single inbound request.

F, indicating outbound system responses which free the terminal keyboard should be displayed. This may result in multiple outbound screen images as the result of a single inbound request.

T, indicating only the final outbound screen image, or transaction image, which precedes the next inbound screen image should be displayed. This will result in a single outbound screen image as the result of a single inbound request. This is the option which will result in a logical user view of an interactive dialogue with an online system.

Attributes Option

The Browse Panel attributes option allows you to specify the type of terminal to be used to display the screen images. You may specify one of the following terminal types:

E, indicating extended data stream color and highlighting attributes are to be displayed. This option requires that the physical device you are using must be an extended 3270 data stream terminal and the trace data set must have been captured from an extended 3270 data stream virtual terminal

B, indicating basic 3270 data stream attributes are to be displayed. You may use this option to see how a trace data set captured with extended attributes would appear on a basic 3270 terminal device.

Browse Panel Commands

CA Enterprise Tester commands available at the Browse Panel are displayed in the active command list. The PF key command assignments may differ from your list based on installation options and your own PF key customization.

- If the Enter key is pressed, the VIEW command is used to initiate the browse and the browse movie is displayed without freeing the terminal keyboard. When the browse has completed, the last browse screen image will be displayed. A step display time interval may be specified on the Options Panel to determine how long each screen image will be displayed on the terminal.
- If the STEP command is used to initiate the browse, the first screen image will be displayed and the browse is stopped, freeing the keyboard for further input or commands.
- The LOCATE command, abbreviated L, may be used to position the browse at a specific screen image. You must specify a screen number with the LOCATE command.

- While viewing the Browse Panel, the END command may be entered to avoid initiating a browse, or to return to the Control Panel.
- The KEYS command, abbreviated K, may be used to display the CA Enterprise Tester Keys Panel. The Keys Panel may be used to assign commands to PF keys. The STEP, VIEW, ZOOM, REVIEW, UP and DOWN commands may be used frequently when moving through browse displays. You should assign these commands to PF keys to simplify command entry.
  - The OPTIONS command, abbreviated O, may be used to display the CA Enterprise Tester Options Panel. The Options Panel supports two options that directly influence the operation of a browse. When the VIEW command is entered to initiate or resume a browse, a movie is displayed that shows the terminal inputs and online system responses that flow on the session being simulated by the browse. The Options Panel provides controls which influence the speed of the movie. The following option setting causes each screen image to be displayed for 1 second before proceeding to the next image.

    ```
    Step display time for session 'movies'  ===> 0100  (SSTH)
    ```

  - Screen images may be displayed using either Zoom Mode or Window Mode. The Options Panel controls the initial screen image display mode. The following option setting displays screen images using Window Mode.

    ```
    Should PLAY, ANALYZE and BROWSE start in Zoom Mode?  ===> NO (Yes or No)
    ```

    The Window Mode display has the advantage of showing the direction of a message, AID key and other information in the panel title. The Zoom Mode display has the advantage of showing the full session screen.

CA Enterprise Tester Session Browse Panel

If the browse is initiated in Window Mode, or Window Mode is invoked with the ZOOM command or key, the CA Enterprise Tester Session Browse Panel will be displayed, as illustrated in the following example.

```
+-------------------------------------------------------------------------------+
| ProTerm Session Browse(7, PT01 <--CD CICS) ---------------------- Port 001 |
| COMMAND ===> SCROLL ===> CSR                                                  |
|---+----1---+----2----+----3---+- RS1.TRACE.PDS(CICSTRC3) #00007 11:34:09:17 |
| S DA(HFTRFILE) ENABLED                                                        |
| STATUS: RESULTS - OVERTYPE TO MODIFY                                          |
| Dat(HFTRFILE) Vsa Clo Ena Rea Upd Add Bro Del Sha NORMAL                      |
| Obj( CICS.HFTR.FILE )                                                        |
|                                                                               |
+-------------------------------------------------------------------------------+
```

In the example, the Session Browse Panel displays the following information:

- The title of the panel, followed by an open parenthesis, (
- The number of the browse panel relative to this simulated session, 7
- The virtual terminal ID used when the session was traced, PT01
- The direction of transmission, indicating it is an outbound screen image, and that the message contained a change direction indicator, <--CD
- An inbound screen image displays the AID key which initiated the transaction preceding the directional arrow, for example, ENTER -->
- The name of the online system used by the session followed by a close parenthesis, CICS
- The current port on which the browse is displayed, Port 001
- The name of the trace or trace list data set; the screen sequence number and the time stamp of the original session obtained from the title line of the traced screen displayed on the scale line:

```
. . . 3----+- RS1.TRACE.PDS(CICSTRC3) #00007 11:34:09:17
```

Session Browse Panel Commands

CA Enterprise Tester commands available at the Session Browse Panel include those listed below. You will find PF key command assignments particularly useful when moving through browse screen image displays.

- The LOCATE command, abbreviated L, may be used to position the browse at a specific screen image. You must specify a screen number with the LOCATE command.
- The VIEW command, abbreviated V, may be used to initiate a browse movie. When the browse has completed, the last browse screen image will be displayed.
- The STEP command, abbreviated STE, may be used to advance the browse a single session step display. An optional number of session steps may be entered.
- The REVIEW command, abbreviated REV, may be used to display a prior session step. An optional number of session steps may be entered.
- The virtual terminal screen image scroll commands, UP, DOWN, RIGHT, and LEFT may be used to view areas of the trace data set screen images which may not be displayed in the window you are viewing.
- The END command, abbreviated E, may be entered to return to the CA Enterprise Tester Control Panel. The END command, entered at the Session Browse Panel, will not end the browse. You may end the browse by entering the END BROWSE command, abbreviated E B port, from the Control Panel.

- The OPTIONS command, abbreviated O, may be used to display the CA Enterprise Tester Options Panel.
- The KEYS command, abbreviated K, may be used to display the CA Enterprise Tester Keys Panel.

Analyzing Traced Sessions

CA Enterprise Tester sessions, session traces and session analyzers may be used to automate regression testing. As explained in Starting Virtual Terminal Sessions and Using CA Enterprise Tester Sessions, CA Enterprise Tester sessions may be used interactively in the same way that real terminal sessions are used to access online systems and enter transactions. As explained in Tracing Session Screens, CA Enterprise Tester sessions that are used interactively to access online systems, may be traced to capture each terminal input entered by a user and the response to each transaction sent by an online system in a trace data set.

Using PSL statements, CA Enterprise Tester scripts may also start and use virtual terminal sessions that look like ordinary terminal sessions to an online system, and may use other PSL statements to read and process screen images that were previously captured in a trace data set when an interactive user session was traced. Scripts that create and use new virtual terminal sessions to repeat transactions read from a trace data set, and that analyze transaction responses by comparing new responses to old responses to identify differences, are called session analyzers, and may be used to perform regression testing automatically.

The CA Enterprise Tester Analyze Panel is used to specify previously traced session data to be analyzed and to start a session analyzer when regression testing is performed interactively. For complete information about CA Enterprise Tester functions used to perform regression testing, session analyzers, how to customize session analyzer processing and how to use session analyzers interactively and in batch, refer to Regression Testing.

CA Enterprise Tester Analyze Panel

The CA Enterprise Tester Analyze Panel, illustrated in the following example, is displayed when the ANALYZE command and an optional port number is entered at the Control Panel. If a port number is not entered, CA Enterprise Tester will default to the next available port.

For example:

```
COMMAND ===> a 1  (and press the Enter key)
```

```
+-------------------------------------------------------------------------------+
| ProTerm Analyze Panel ------------------------------------------- Port 001 |
| COMMAND ===>                                                                  |
|                                                                               |
| ENTER TO VIEW I/O PF15 END ANALYZE REQUEST K KEYS PANEL                       |
| PF23 STEP TO SCREEN 1 LOG LOG PANEL O OPTIONS PANEL                           |
|                                                                               |
| Specify trace data set or trace list data set to be analyzed:                 |
|                                                                               |
| Data Set Name ===> 'UID1.PROTERM.TRACE'                                       |
| Member Name ===> VENDRADD ( Blank or pattern for member selection list )      |
| ( Select codes: S Select, B Browse, E Edit )                                  |
| Specify Script Library to access private session analyzer scripts: **         |
|                                                                               |
| Script Library ===> 'UID1.PROTERM.SCRIPT'                                     |
| Mask Script ===> MASKAP ( * or pattern for member selection list )            |
|                                                                               |
|                                                                               |
|                                                                               |
| Analyzer Script ===> PTESAFE ( PTESAFE Fast Fixed Event Session Analyzer )    |
| ( PTESAVE Timed Variable Event Analyzer )                                     |
| ( PTESAVED Delayed Variable Event Analyzer )                                  |
| ( Blank or pattern for member selection list )                                |
|                                                                               |
| ** Script Library is searched before PTE.R310.SCRIPT                          |
+-------------------------------------------------------------------------------+
```

The Analyze Panel is used to specify previously traced session data to be analyzed and to start a session analyzer when regression testing is performed interactively.

Short messages issued while using the Analyze Panel are displayed on the right side of the Analyze Panel title line.

A standard ISPF command entry area on the second line of the panel permits entry of CA Enterprise Tester commands or other standard ISPF or TSO commands.

The third line of the panel is the standard ISPF long message display area. A long message is displayed when you press the PF key to which you have assigned the HELP command, (usually PF1), when a short message is displayed, further describing an error or unusual condition, or giving other additional information.

Trace Data Set and Member Name

Analyze Panel trace Data Set Name and Member Name data entry areas are used to specify the traced session data to be analyzed. The data set name area is a 46-character area, permitting entry of a fully-qualified 44-character data set name, enclosed in quotes. A user-specified prefix is added to the data set name if the full data set name is not entered in quotes. For a description of prefix options, please refer to Options and Controls.

The data set specified may be either a sequential data set, or a partitioned library. If a partitioned library is specified, a member name may also be specified.

If you do not specify a member name in a library, or if you specify a pattern using either the asterisk, * , or the percent sign, % , a library directory or member selection list is displayed when you press the Enter key. Select codes, B, E or S may be used to Browse, Edit or Select a listed member. If a command is pending when the list is displayed, it will remain pending, and executed if a member is selected.

The data set or library member specified may be either a CA Enterprise Tester session trace data set captured using the trace FORMAT option, or may be a trace list containing control statements that identify one or more trace data sets to be analyzed, and analyzer processing options. Trace lists and analyzer options supported by the distributed session analyzer, PTESA, are explained in The CA Enterprise Tester Session Analyzer.

Script Library and Scripts

The Script Library data entry area is used to specify the data set name of the partitioned data set (PDS) or library containing a mask script, a user-customized session analyzer script, or any other user-supplied scripts used by a session analyzer to customize processing.

If a Script Library is not specified or if needed scripts are not found in the user-specified library, the CA Enterprise Tester system script library is searched automatically. The systems programmer who installed CA Enterprise Tester defines the system script library. The name of the library data set is displayed on the bottom line of the Analyze Panel.

The Mask Script data entry area is used to specify a library member name of a mask script. A mask script is an optional user-written script that may be used to identify and suppress unwanted, predictable session mismatches automatically, for example, displayed dates and times. A mask script may also be specified in a trace list or in a particular analyzer script. Refer to Masking Dates, Times, and Other Data for complete information about mask scripts and how to create them.

Member Name pattern characters, either the asterisk, * , or the percent sign, %, may be entered to display a member selection list of the user-specified script library. While viewing a member selection list, select codes, B, E or S may be used to Browse, Editor Select a listed member.

The Analyzer Script data entry area is used to specify the library member name of a session analyzer script to be used to process the trace data set or trace list.

The analyzer scripts listed on the Analyze Panel use different synchronization options supported by the distributed session analyzer, PTESA. PTESAFE may be used to analyze sessions that respond predictably at maximum speed. PTESAVE and PTESAVED use synchronization options that allow sessions that may respond unpredictably to be analyzed reliably by allowing time for an unpredictable system to respond.

Simple user-written analyzer scripts that link to PTESA may be used to upgrade analyzed trace data sets automatically while they are being analyzed, to specify user-written scripts that may be used to customize certain analyzer processing, or to specify other preferred

analyzer options. Please refer to The CA Enterprise Tester Session Analyzer and Session Analyzer Options for more information on customizing analyzers.

Blanks or member name pattern characters may be entered as the analyzer script name to display a member selection list of the system script library if a user-specified script library is not entered.

While viewing a member selection list, select codes, B, E or S may be used to Browse, Edit or Select a listed member.

Analyze Panel Commands

CA Enterprise Tester commands available at the Analyze Panel are displayed in the active command list. The PF key command assignments illustrated in the Analyze Panel example may differ from your list based on installation defaults and your own PF key customization.

- If the Enter key is pressed after specifying session analyzer input, the default command, VIEW, processes an entire session, displaying each session step screen image, and stopping only when a mismatch is not resolved by the session analyzer script.
- As an alternative to the default VIEW command, an analysis may be started using the STEP command or by pressing a PF key to which the STEP command is assigned. The STEP command key may be pressed at each display to proceed to the next session step.

- When the STEP command is used, and a session screen is displayed using either Zoom Mode or Window Mode, the ZOOM command, or the PF key to which the ZOOM command has been assigned, may be used to toggle between the two display modes.
- While stopped at a session step, the VIEW command may be entered at any time to proceed automatically.
- While viewing the Analyze Panel, the END command may be entered to return to the CA Enterprise Tester panel at which the ANALYZE command was entered.
- The LOG command may be used to display the CA Enterprise Tester Log Panel and to open a specified log dataset to capture test results. Test results are captured only if a log active while the session analyzer is active.
- The KEYS command may be used to display the CA Enterprise Tester Keys Panel. The Keys Panel can be used to assign commands to PF keys. The STEP, VIEW, ZOOM, REVIEW, ROLL, UP and DOWN commands may be used frequently while analyzing sessions. You should assign these commands to PF keys to simplify command entry.
- The OPTIONS command is used to display CA Enterprise Tester Options Panels. The Options Panel(s) support options that directly influence the operation of a session analyzer.
  - When the Enter key is pressed to start a session analyzer or the VIEW command is entered to resume a session analyzer, a movie is displayed that shows the terminal inputs and online system responses that flow on the session being re-created by the session analyzer. The Options Panel step display time controls the speed of the movie. The following option setting causes each screen to be displayed for 1 second before proceeding to the next session step.

    ```
    Step display time for session 'movies'  ===> 0100  (SSTH)
    ```

  - Session steps may be displayed using either Zoom Mode or Window Mode. An Options Panel setting controls the initial session display mode. The following option setting displays screens using the Session Panel in Window Mode.

    ```
    Should PLAY, ANALYZE and BROWSE start in Zoom Mode?  ===> NO (Yes or No)
    ```

    The Window Mode display has the advantage of showing the direction of a message, AID key and other information in the panel title. The Zoom Mode display has the advantage of showing the full session screen.

Interacting with Session Analyzers

A specified analyzer script is started when a valid trace data set or trace list is entered at the Analyze Panel using the Enter key, VIEW command or STEP command. If the Enter key or VIEW command is used to start an analyzer script, the new session created by the analyzer is displayed as a session movie that proceeds until a mismatch or other problem is detected. The STEP command or PF key may be used to start an analyzer script and may be used to advance analyzer processing to the next inbound session step or system response, one screen at a time, until a mismatch or other problem is detected by the session analyzer.

The CA Enterprise Tester Monitor Panel is displayed when a mismatch or other problem is detected. Messages displayed in the Monitor Panel Message Log explain the mismatch or other problem detected, and options available to the user.

Analyzing Session Mismatches explains CA Enterprise Tester displays and functions that may be used while an analyzer is stopped, to view, analyze or correct a problem, and how to resume analyzer processing.

Script Monitoring and Interactive Debug

The CA Enterprise Tester Interactive System Monitor and it's associated Monitor Panel support many different functions in the analysis and control of script processing.

- CA Enterprise Tester scripts may display messages to the terminal user via the Monitor Panel Message Log. Messages may be system generated or generated by the script.
- When a script is played under the control of the Interactive System, a Script Language PAUSE statement will interrupt script processing and display the Monitor Panel. The user may then examine the message log, enter Interactive System commands and perform other tasks relating to debugging the script.
- Messages detailing session mismatches detected by a CA Enterprise Tester session analyzer are displayed using the Monitor Panel Message log.
- Errors messages describing syntax errors in executed scripts and errors that occur during script execution are presented in the Message Log.
- The Monitor Panel provides status messages, session I/O statistics, script execution statistics, and may be used while a script is executing in the TSO background to monitor script execution.
- The Monitor Panel shows the next or current script statement being executed and provides commands and stop capabilities that may be used to control script execution and debugging.

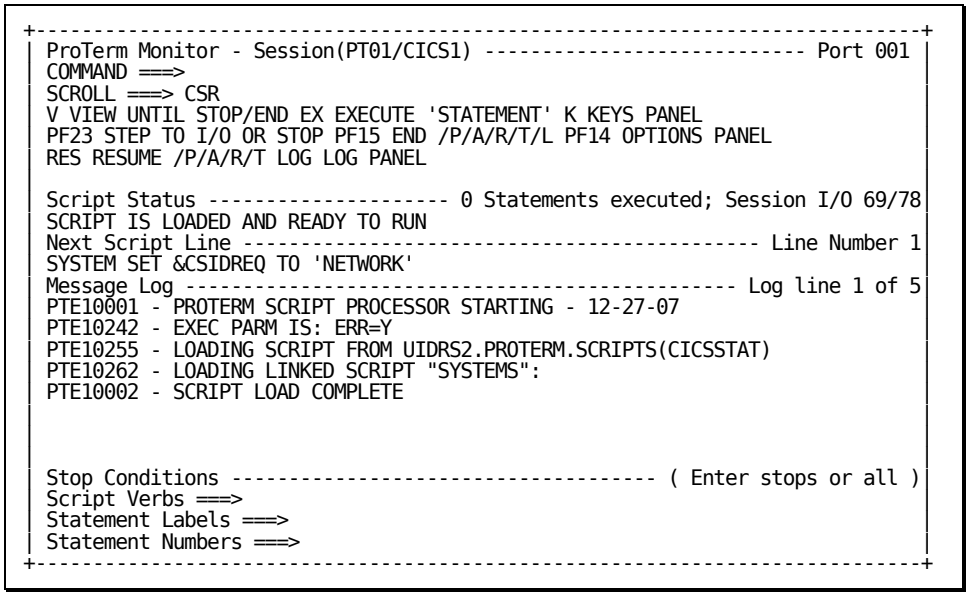
- The Monitor Panel is displayed when the MONITOR command, abbreviated M, is entered at the Play Panel to start a script.
- The Monitor Panel is displayed when a script is active on a port and the MONITOR command is entered at the Control Panel or Session Panel.
- When a VIEW or STEP command is entered to process script-driven events, the Monitor Panel is displayed if an error occurs, if a specified script statement stop is matched, or if a script stops by executing a Script Language PAUSE or WTOR statement, before the VIEW or STEP command completes it's normal function. The Monitor Panel contains messages that explain the exact reason for stopping the script.
- The Monitor Panel is displayed if the END command is entered while reviewing session screens or while using a session that is on a port that also has a stopped script.

CA Enterprise Tester Monitor Panel

The CA Enterprise Tester Monitor Panel is displayed when the MONITOR command is entered at the Play Panel to start a script, or at the Control Panel for a port on which a script is active. Other conditions such as errors or specific Script Language coding may also result in a Monitor Panel display.

The data displayed on the Monitor Panel depends on many factors relating to the control and execution of the script, and the status of the virtual terminal session with the online system.

The following example illustrates one example of a Monitor Panel display.



Note the following points about the Monitor Panel:

- The panel title shows the partners in a CA Enterprise Tester session if a session is active. The short message area (top/right) shows the current port number when a message is not displayed. The Monitor Panel has a two-line command field and a standard scroll amount field.
- The Monitor Panel command list is dynamic. The commands listed at any point in time are commands that are acceptable given the current situation. For example, RESUME is listed in the sample display because the played script is stopped. The RESUME command causes a stopped script to resume execution. When the RESUME command is entered, it is removed from the command list and is replaced by the STOP command which is appropriate while a script is executing. Similarly, the SESSION command, used to select an active session, is listed only while a CA Enterprise Tester session is active on the port being monitored.
- The Script Status lines shows the current status of a script. The status message indicates if the script is waiting at stop, waiting at error, or executing. Statistics presented in the Script Status area show the number of script statements that have been executed, and the number of input and output messages that have been sent on an active session. These numbers may be used to gauge the progress of the script being monitored.
- The Next Script Line area shows the statement number and statement that will be executed next if a stopped script is resumed. In the illustration, the script is not currently executing. When script execution is resumed, the Next Script Line area becomes a Current Script Line area. The Current Script Line area shows the statement that was executing when the Monitor Panel was last displayed. The Enter key updates the display.
- During the execution of a script, status messages, error messages and messages created by a script using the Script Language LOG statement, are written to a script log. The Monitor Panel Message Log area is a scrollable area through which the script log may be viewed. The Message Log contains the messages that would be written to the PTELOG DD statement if the script were executed as a batch job step.
  - If the LIST option is specified via the Play Panel EXEC PARM field, then a listing of the played script is placed in the Message Log while the script is being loaded for execution.
  - If the FLOW option is specified via the Play Panel EXEC PARM field, or if a script issues a Script Language FLOW statement, then each statement is listed in the Message Log just prior to being executed.
  - Messages documenting key events and errors encountered during script execution are listed in the Message Log.
  - Messages generated by Script Language LOG, WTO and WTOR statements are listed in the Message Log.

  - UP, DOWN, LEFT, RIGHT and LOCATE commands support Message Log scrolling.
  - When the Enter key is pressed, if messages have been added to the Message Log, the log presentation area is automatically repositioned to the bottom of the log so that the most recent messages are displayed.

- The Monitor Panel Stop Conditions area allows stops to be placed at specified points within a script. The following example shows valid Stop Condition specifications:

```
Script Verbs  ===> TYPE END
Statement Labels  ===> SETUP LOOP
Statement Numbers ===> 7 25
```

The number or combination of stop conditions is limited only by the size of the Monitor Panel's Stop Condition fields. Stops may be modified at any time. The word all may be entered in any field. ALL, specified in the Script Verbs or Statement Numbers fields, causes a script to be stopped prior to the execution of each script statement. ALL, specified in the Script Labels field, causes a script to be stopped prior to the execution of each labeled statement.

Monitor Panel Commands

Monitor Panel primary commands support monitoring, control and debugging of played CA Enterprise Tester scripts.

- CA Enterprise Tester session traces and/or script recordings may be initiated and terminated via the Monitor Panel using TRACE, RECORD, END TRACE, and END RECORD commands.
- The STOP and RESUME commands may be used to stop and resume all features that have been initiated on a port. STOP and RESUME commands may optionally specify PLAY, RECORD or TRACE, to suspend the designated feature only. For example:

```
COMMAND ===> sto t
```

or

```
COMMAND ===> res t
```

- The SESSION command may be used to select and use a CA Enterprise Tester session that is active on the port being monitored. If a script is executing when the SESSION command is entered, then the script is stopped prior to displaying the session. The session is presented in either the Window Mode Session Panel or Zoom Mode session display. The session may be used interactively. It is the user's responsibility to consider the impact of using a session that was being used by a stopped CA Enterprise Tester script.

- The VIEW command may be used to start or resume a script. Each online system screen image or session step is displayed if a non-zero step display time interval has been specified on the Options Panel. The script will continue displaying each online system screen image until the script ends or a session event or error occurs which is not anticipated by script logic. If a script is stopped when a VIEW command is entered, then the script is resumed.
- The STEP command can be used to resume script execution and to wait until a message is sent or received by the executing script via the CA Enterprise Tester virtual terminal that is attached to the port being monitored. When a message is detected, the CA Enterprise Tester session screen is presented in either the Window Mode Session Panel or Zoom Mode session display. The Session Panel title identifies the display as being a session step as opposed to a normal session display. The session can not be used interactively while at a session step. However, while at a session step, the SESSION command may be used to advance a script to the end of the current TYPE or INVITE statement that is driving the session I/O, after which the session is presented for interactive use. Successive inbound and outbound session messages may be viewed as they occur by repeated use of the STEP command.
- The VIEW or STEP commands, particularly when assigned to a PF key to expedite repeated command use, can be used in conjunction with Monitor Panel Stop Conditions including the ALL option, to execute a script one statement at a time or stopping only at specific points in a script. Of course, errors always stop a script for diagnostic or corrective user action.
- The EXECUTE command provides a powerful debugging tool. The command allows CA Enterprise Tester Script Language statements to be immediately executed by the Script Language processor. The EXECUTE command may be entered at any time. If a script is executing when an EXECUTE command is entered, the script is stopped until the EXECUTE command has been processed, then the script is resumed. If a script is at a session step when an EXECUTE command is entered, then the script statement driving the session I/O is completed prior to processing the EXECUTE command. The following examples illustrate a few of the many possible uses of the EXECUTE command.
  - Script execution speed may be dynamically altered:

    ```
    ex 'think 100'
    ```

  - The next statement to be executed may be dynamically altered:

    ```
    ex 'go to 27'
    ```

  - Variable memory may be dynamically corrected or altered:

    ```
    ex "set &vfield to 'any value' "
    ```

  - Variable memory may be displayed via the Monitor's Message Log:

    ```
    ex 'log "/" &vfield "/" &row "/" <24 1 40> "/" '
    ```

  - You may try out Script Language statements just to see how they work:

    ```
    ex 'set &hex from hex "A1" log &hex'
    ```

- The REVIEW and STEP commands may be used as described in Interactive Session Review to re-display past session steps. The REVIEW command may be entered at any time, provided that session messages have previously been processed on the port. If the REVIEW command is entered while a script is executing, then the script is stopped prior to processing the REVIEW command. The REVIEW and STEP commands may specify a number of session steps to back up or advance more than one session step at a time.
- While monitoring the execution of a played script via the Monitor Panel, the ROLL command may be entered to display the next CA Enterprise Tester port that has an active session or played script. If no higher numbered ports qualify, the ROLL command selects the first port that has an active session or script. If a script is active on the selected port, then the port is displayed via the Monitor Panel, otherwise, the port's session is displayed in either the Window Mode Session Panel or Zoom Mode session display. Repeated use of the ROLL command, while multiple scripts are executing during a load test, allows detailed monitoring of all active scripts.
- The END command may be entered to exit the Monitor Panel display and return to the Control Panel to start or monitor CA Enterprise Tester functions on other ports. Exiting the Monitor Panel display does not change the status of the script being monitored.
- An END PLAY command, abbreviated e p, may be entered to immediately terminate script execution. When script execution ends, if a session is active on the port being monitored, then the session is displayed in either the Window Mode Session Panel or Zoom Mode session display, otherwise, the Control Panel is displayed.
- The KEYS command provides access to the Keys Panel to view and alter function of the PF keys.
- The OPTIONS command provides access to the Options Panel to view and alter CA Enterprise Tester options.
- The LOG command may also be entered to start or resume a CA Enterprise Tester log.

Script Logs and Diagnostic Traces

When CA Enterprise Tester scripts are executed interactively using the Analyze Panel or the Play Panel, script logs are kept in memory and may be viewed using the Monitor Panel. When a script ends, the log is purged. The Interactive System Log Facility may be used to capture script logs in a data set.

Other information may also be captured in an Interactive System log. For example, if a message sent by an online application is rejected by a terminal, then the message contains an invalid 3270 command or order, or violates some rule of session protocol. A data stream error is usually seen by a terminal user as a PROG470 or other code displayed on the status line of the terminal screen. Because the terminal is unable to process the error, the invalid message itself is not seen, making it difficult to determine the specific cause of the error.

If an invalid online system message is sent to a CA Enterprise Tester virtual terminal, the message is rejected as it would be on a real terminal. However, if a CA Enterprise Tester log is active when the error occurs, a message that describes the error and a dump of the invalid online system message is written to the log. The dump may be used to identify the cause of the error.

The Log Facility may also be used to capture a dump of session RU data. RU means request unit, and is an SNA term that is used to refer to a unit of information that is exchanged on a session. A session RU dump is a very low level session trace that contains messages that describe session and logical unit states and that contains a dump of all request units that flow on a session. A request unit dump is a separate feature from a script log and does not depend on a script being active on a port.

An Interactive System log is not started on a particular port, but rather is a central feature. When a log is active, messages logged by other CA Enterprise Tester functions that are active on any number of Interactive System ports are captured in chronological order in a single log data set. When messages are logged concurrently from more than one port, extra messages identify the port from which subsequent messages originated.

When a multiple session online system test is conducted by playing multiple scripts concurrently on multiple Interactive System ports, a single log data set may be used to capture all script logs. When data is written to JES, it is possible to view the log while it is active by using ISPF split-screen to access SDSF or another JES access tool to browse the data set.

CA Enterprise Tester also provides a utility function which will sort the central log messages into order by port number. For more information on CA Enterprise Tester Utilities, refer to CA Enterprise Tester Utilities.

An Interactive System log may be started by entering the LOG command in the command field of most CA Enterprise Tester panels. The LOG command displays the CA Enterprise Tester Log Panel, illustrated in the Log Panel example.

The LOG command is displayed in the CA Enterprise Tester Control Panel command list. Once a log is active, the LOG command is no longer displayed on the Control Panel command list, and the message, Log Active, is displayed in the lower right corner of the panel.

CA Enterprise Tester Log Panel

```
+-------------------------------------------------------------------------------+
| ProTerm Log Panel ------------------------------------------------------------ |
| COMMAND ===>                                                                  |
|                                                                               |
| ENTER TO START                                                                |
| PF15 END LOG REQUEST                                                          |
|                                                                               |
| Specify target data set, data set and member, or SYSOUT=class,DEST=destname:  |
| Data Set Name ===> 'UIDRS2.PROTERM.LOG'                                       |
| Member Name ===> ( Blank or pattern for member selection list )               |
| ( Select codes: S Select, B Browse, E Edit )                                  |
|                                                                               |
| Session RU dump? ===> NO ( Yes or No )                                        |
|                                                                               |
| Protect target? ===> NO ( Yes or No to replace data )                         |
|                                                                               |
| Share PDS? ===> NO ( Yes or No )                                              |
|                                                                               |
+-------------------------------------------------------------------------------+
```

The Log Panel is used to initiate an Interactive System log, and to enter options and specifications which control how the log is to be captured.

The panel name is displayed on the left, and the upper right corner is used for standard ISPF short message display.

A standard ISPF command entry area on the second line of the panel permits entry of CA Enterprise Tester commands as well as ISPF commands such as SPLIT and SWAP, and execution of TSO commands.

The third line of the panel is the standard ISPF long message display area. A long message is displayed when you press the PF key to which you have assigned the HELP command, (usually PF1), when a short message is displayed, further describing an error or unusual condition, or giving other additional information. Not all short messages will have additional long messages.

The log is started when the Enter key is pressed at the Log Panel.

Data Set Name Parameters

The name of a target data set for storage of log data may be entered in this area of the Log Panel. The data entry area is a 46 character area, permitting entry of a fully-qualified 44 character data set name enclosed in quotes. The data set may be:

- A sequential data set with data set characteristics RECFM=FBA and LRECL=151.
- A partitioned data set name and member name with data set characteristics RECFM=FBA and LRECL=151;
- A JES SYSOUT class and optional destination name.
- At entry to the Log Panel, if the data set name area is null, the name is primed according to a system installation option. If the user leaves it there and specifies a member, then the data set is created automatically when the log is started.

  If the message, Dataset not found is displayed in the upper right hand corner of the screen, when you attempt to start the log, this means CA Enterprise Tester was unable to automatically allocate the data set for you. You may use ISPF's Data Set Utility function, (usually Option 3.2) to allocate the dataset. The only CA Enterprise Tester requirements are that the data set must be allocated with RECFM=FBA and LRECL=151.

If you do not specify a member name in a PDS, or if you specify a pattern using either the asterisk, * , or the percent sign, % , a directory will be displayed when you press the Enter key, from which a member may be selected. You may also browse or edit members from the directory display. If a command is pending when the directory is displayed, it remains pending, and will be executed when a member is selected.

Previous releases of CA Enterprise Tester supported a separate data entry area for specifying only a SYSOUT class. You may now enter SYSOUT class in the data set name data entry area.

Session RU Dump Option

The request unit dump feature is a diagnostic aid that documents virtual terminal sessions and data streams at the lowest possible level.

- If YES is specified, then CA Enterprise Tester will initiate a the request unit dump.
- NO, the default, specifies that no request unit dump will be initiated.

Protect Target Option

CA Enterprise Tester provides options to permit you to specify whether or not existing data in either a sequential data set or a member of a PDS should be overlaid with new log data.

- If a protect option of YES is specified, then CA Enterprise Tester will not initiate the log activity which would overlay the existing data, but instead issue a message warning you that the member exists or that the data set is not empty.
- If a protect option of NO is specified, then the existing data will not be protected and will be replaced by the new log data.

The initial value for this option, displayed when you enter the panel, may be specified on the second page of the CA Enterprise Tester Options Panel. For a complete description of CA Enterprise Tester options, refer to Options and Controls.

Share PDS Option

The PDS share option may be used to allow multiple users to write to the same PDS at the same time.

- If YES is specified, CA Enterprise Tester will provide dynamically allocated work data sets for the actual recording of log data, and copy the data to the target PDS when the log is ended.
- If NO is specified, exclusive control will be maintained at the PDS level.

Directory statistics will be maintained for the PDS based on the options specified on Page 2 of the CA Enterprise Tester Options Panel.

Log Panel Commands

CA Enterprise Tester commands available at the Log Panel include those listed below. The PF key assignment illustrated in the Log Panel example may differ from your own based on system installation options and your customization.

- When valid required data has been entered in the Log Panel and the Enter key is pressed, the log data set is opened for subsequent logging activity. After initiating the log, the panel from which the LOG command was entered is re-displayed. The LOG command is not displayed on the Control Panel command list when a log is active, and the message, Log Active, is displayed in the lower right corner of the panel.

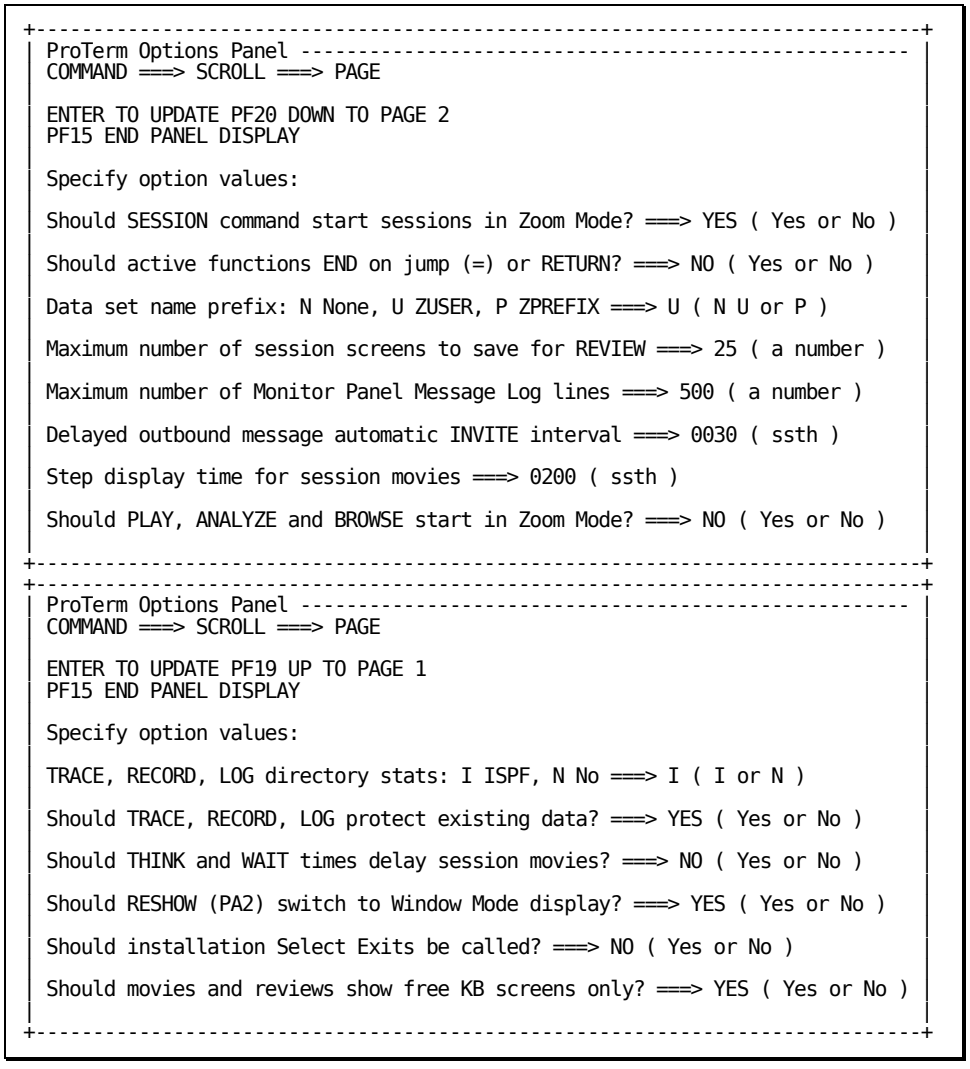
  The LOG command may be used to display the Log Panel, even when a log is already active. This allows you to turn the request unit logging feature on or off without ending and restarting the log. When the Log Panel is displayed and a log is already active, the Enter key may be pressed to resume logging without interruption.

- The END command may be entered at the Log Panel to terminate logging and to close the log data set. A log may also be ended while using other CA Enterprise Tester panels by entering the END LOG command, abbreviated e l.

Options and Controls

The CA Enterprise Tester Interactive System provides options and controls that allow each user to control how CA Enterprise Tester is to perform certain tasks and to specify resource limits to be observed by CA Enterprise Tester functions.

Options and current option settings may be viewed and altered using the CA Enterprise Tester Options Panel. The Options Panel is displayed when the OPTIONS command, abbreviated O, is entered at other CA Enterprise Tester panels. The Options Panel is illustrated the following example.



CA Enterprise Tester Options Panel

The CA Enterprise Tester Options Panels provide display of current option settings which may be changed in the data entry areas provided. The options displayed initially on the CA Enterprise Tester Options Panels are specified when CA Enterprise Tester is installed and may differ from those illustrated. Options may be changed by keying over the current setting where indicated and by pressing the Enter key. Option settings apply to all CA Enterprise Tester functions and ports, except where an active function has already exceeded a newly specified resource limit. The following sections describe each option and option choices.

- Should SESSION command start sessions in Zoom Mode?

  CA Enterprise Tester presents sessions with online systems in one of two display modes, either framed in the CA Enterprise Tester Session Panel or Window Mode, which enables a session to be viewed in ISPF split-screen mode, or in full-screen Zoom Mode, through which an entire session screen may be viewed without ISPF panel framing.

  When new sessions are started by entering the SESSION command at the Control or Play Panel or when a session is started using the CA Enterprise Tester System Menu, then the initial session display mode for the session is determined by this option setting. When YES is specified, new sessions are initially presented using the full-screen Zoom Mode session display. When NO is specified, new sessions are initially presented using the CA Enterprise Tester Window Mode.

- Should active functions END on jump (=) or RETURN?

  The jump control allows you to specify whether active CA Enterprise Tester functions are to be automatically terminated when you attempt to exit CA Enterprise Tester when an END command if entered at the CA Enterprise Tester Control Panel, or when a RETURN or jump, (=), command is entered at any CA Enterprise Tester panel. When YES is specified, and you enter a command to exit CA Enterprise Tester, active sessions, scripts, recordings, traces and browses are automatically ended before CA Enterprise Tester returns to ISPF. When NO is specified, and CA Enterprise Tester functions are active, CA Enterprise Tester will discard the RETURN or jump (=) command, display the CA Enterprise Tester Control Panel, and display a message requesting that active functions be ended before exiting CA Enterprise Tester. You may then enter the correct END commands to terminate active functions. A jump option setting of NO protects the user from accidental termination of active functions.

- Data set name prefix: N=none, U=ZUSER, P=ZPREFIX

  The data set name prefixing option allows you to specify the format in which data set names will be entered in data set name fields provided on other CA Enterprise Tester panels. When N is specified, fully qualified data set names must be entered. When U is specified, and a data set name is entered without quotes, your TSO log on identifier, the ISPF dialog variable, ZUSER, will be added as the high level node of the data set name. When P is specified, then standard TSO data set name prefixing, using your specified prefix in the ISPF dialog variable, ZPREFIX, will apply. (The ZPREFIX variable value may be changed using the TSO PROFILE command and the PREFIX(value) option of the command.) A data set name enclosed in quotes will be accepted as is.

- Maximum number of session screens to save for REVIEW

  While CA Enterprise Tester sessions are active, recent screen images are automatically saved for subsequent review. Saved screens may be accessed using the REVIEW command. The review limit option allows you to specify the number of most-recent screens to be saved. The limit may be a number between 0 and 99, and specifies the number of screens to be saved for each concurrently active session. Saved screens require approximately 4K bytes of main memory for each saved screen and associated control information. While conducting load tests that involve a large number of sessions, a small review screen limit may be specified to conserve main storage. While conducting single session regression tests, a larger number may be specified to allow review of a larger number of previous session steps.

- Maximum number of Monitor Panel Message Log lines

  While using session analyzers and other played scripts interactively, a message log is kept in main storage. The Message Log is displayed on the Monitor Panel. The log limit control may be used to specify the maximum number of most-recent log lines to be kept in main storage. When 0 is specified, lines are not limited. When a number is specified, and the number of logged lines exceeds the limit, older log lines are discarded to make room for recently logged lines. Up to the number of log lines specified will be saved for each concurrently executed script. Like the review control, the log control may be used to conserve main storage when multiple scripts are executed concurrently to perform various types of load tests.

- Delayed outbound message automatic INVITE interval

  Some online systems send delayed messages that follow a message that causes the terminal keyboard to be freed. When a message that frees the terminal keyboard is presented using the CA Enterprise Tester Session Panel in Window Mode, a delayed online system message can not be presented until the ISPF display manager returns control to CA Enterprise Tester. The Zoom Mode session presentation uses native TSO display services and is not subject to this difficulty.

You will recognize the problem when an expected message is not received. An INVITE command may be entered manually to receive the delayed message. Alternatively, to minimize the need to manually check for delayed messages, you may specify a time interval that controls automatic invite processing. The time is expressed in seconds, tenths and hundredths of seconds. When a non-zero delayed message invite interval is specified, CA Enterprise Tester will not free the terminal for the specified interval, allowing a delayed message to be displayed without the need to manually enter the INVITE command. A small time interval (ie., less than 1 second) resolves the problem for most online systems without a noticeable impact on response time.

- Step display time for session 'movies'

  While using a session analyzer, playing CA Enterprise Tester scripts, browsing a trace data set or trace list, or while reviewing session steps, the STEP, VIEW and REVIEW commands may be used to move ahead or back up multiple session steps. A session step is a screen image representing a terminal input or an online system response. You may specify a non-zero step display time to request that when a STEP, VIEW and REVIEW command specifies multiple session steps, that each session step be displayed for the specified time interval before automatically moving ahead or backing up to the next session step. When zero is specified, intermediate session steps are not displayed. The time interval is expressed in seconds, tenths and hundredths of a second.

- Should PLAY, ANALYZE and BROWSE start in Zoom Mode?

  CA Enterprise Tester presents sessions with online systems in one of two display modes, either framed in the CA Enterprise Tester Session Panel or Window Mode, which enables a session to be viewed in ISPF split-screen mode, or in full-screen Zoom Mode, through which an entire session screen may be viewed without ISPF panel framing. When a session is started by a session analyzer or other played script, or simulated by a trace browse, then the initial session display mode is determined by this option setting. When YES is specified, screen images are initially presented using the full-screen Zoom Mode session display. When NO is specified, screen images are initially presented using the CA Enterprise Tester Window Mode.

- TRACE, RECORD, LOG directory stats: I=ISPF, N=No

  When CA Enterprise Tester creates or modifies a member in a PDS as a result of trace, record or log activity, ISPF directory statistics will be generated if you specify I as this option. If you specify N, no directory statistics will be generated.

- Should TRACE, RECORD, LOG protect existing data?

  This option will permit you to specify whether or not existing data in either a sequential data set or a member of a PDS should be overlaid with new data as a result of trace, record or log activity. If a protect option of YES is specified, then CA Enterprise Tester will not initiate the trace, record or log activity which would overlay the existing data, but instead issue a message warning you that the member exists or that the data set is not empty. If a protect option of NO is specified, then the existing data will not be protected and will be replaced by the newly created data. The value specified for this option will be used to prime the corresponding option on the Trace Panel, the Record Panel and the Log Panel.

- Should THINK and WAIT times delay session 'movies'?

  Script Language THINK and WAIT statements instruct a script to delay execution of the next statement by a specified time interval. While executing session movies by playing scripts, you may instruct the Interactive System to ignore Script Language THINK and WAIT time intervals by setting this option to NO. If you specify an option of YES, CA Enterprise Tester will wait the specified THINK and WAIT time intervals, even if you are viewing the movie using the STEP command.

- Should RESHOW (PA2) switch to Window Mode display?

  If this option is set to YES, entry of the PA2 key will cause CA Enterprise Tester to re-display the screen image in Window Mode. This is a particularly important feature when your application requires the use of all PF keys. Since no PF key is available to use for the ZOOM command, you must use the PA2 key to switch from Zoom Mode to Window Mode to have access to a command entry area. If this option is set to NO, CA Enterprise Tester will stay in Zoom Mode, and entry of a real or simulated PA2 key will invoke only a TSO RESHOW function. For a complete description of how the CA Enterprise Tester Interactive System interfaces with TSO and ISPF/PDF in processing attention identifier keys, refer to Starting Virtual Terminal Sessions.

- Should installation Select Exits be called?

  CA Enterprise Tester provides installation options which enable user-supplied code to be executed as a customized interface with CA Enterprise Tester. This type of processing is referred to as Select Exits. CA Enterprise Tester Select Exits are optional user-written ISPF dialogs that are executed when CA Enterprise Tester is first invoked or when a user enters a SESSION, PLAY, ANALYZE, BROWSE, RECORD, TRACE or LOG command to display a CA Enterprise Tester panel. If this option is set to YES, the Select Exit processing is enabled. If this option is set to NO, CA Enterprise Tester will function as distributed, and Select Exit processing will be bypassed.

- Should movies and reviews show free KB screens only?

  This option allows you to specify the type of outbound screen image responses from the online system you want displayed during a script-driven session movie or saved for session review displays.

  A session step is a screen image representing a terminal input or online system response. Although a user may enter any number of keystrokes, including Tab keys and Erase EOF keys, all data is combined to form a single inbound screen image to represent the user view of what has been transmitted to the online system. All terminal input or inbound screen images are displayed in a session movie or review.

  Online systems, however, may transmit more than one system response forming the user view of the final output screen image. You may specify one of the following forms for outbound screen images.

  When Yes is specified, only outbound screens that free the terminal keyboard are saved for review, and only outbound screens that free the terminal keyboard are displayed while viewing a session being driven by a CA Enterprise Tester script.

  When No is specified, screens reflecting all separate outbound messages, (chains), are saved for review and are displayed while viewing sessions being driven by CA Enterprise Tester scripts.

  This option will permit viewing of second-level movies, that is, session scripts which logon to TSO, then CA Enterprise Tester again, then create sessions with other systems. When the script is played, the second-level of the script-driven sessions will be displayed.

  This option does not control the granularity of session browses and is temporarily overridden while using Session Analyzers so that the granularity of session and trace reviews is similar. The granularity of session and trace reviews while using Session Analyzers is controlled by the parameters used on TRACK statements used by the session analyzer, PHYSICAL, LOGICAL or TRANSACTION.

Option Panel Commands

CA Enterprise Tester processing and commands available at the Options Panel include the following list. The PF key command assignments illustrated in the Options Panel examples may differ from your list based on installation options and your own PF key customization.

- If the Enter key is pressed, CA Enterprise Tester will immediately update any changes to the option settings and they will apply to subsequent functions, except where an active function has already exceeded a newly specified resource limit.
- The DOWN command, or the PF key to which the DOWN command has been assigned, may be entered on Page 1 of the Options Panel to scroll to Page 2.

- The UP command, or the PF key to which the UP command has been assigned, may be entered on Page 2 of the Options Panel to scroll to Page 1.
- After reviewing or altering Options Panel settings, the END command may be used to return to the panel from which the OPTIONS command was entered.

Configuring Virtual Terminals

Online systems support sessions with terminals that have a variety of features. Some online applications may have dependencies on the type of terminal being used for a session. For example:

- Online applications may take advantage of available terminal features by enhancing displays when particular features are available.
- Some online applications may be designed to operate correctly only when used on particular terminal models.
- Online systems and applications should not send commands and orders to a terminal that uses features that are not supported by the terminal.

CA Enterprise Tester provides features that may be used to test the compatibility of online system software with various types of terminals, and that allow online system software that was designed for use only on a specific terminal model, to be tested while using a physical terminal which is a different terminal model than the one for which the application was designed.

Every CA Enterprise Tester session is a connection between a virtual terminal and an online system. When a CA Enterprise Tester session is started, a virtual terminal is allocated from a pool of terminals. When a virtual terminal is allocated, it is configured to support selected default and alternate screen sizes and to support either the basic or extended 3270 data stream. An online system sees only the features of the virtual terminal that is used for a session and not the features of the terminal being used to access TSO and CA Enterprise Tester.

By default, when a session is started using the Interactive System SESSION command or System Menu, terminal features are selected to be compatible with the terminal being used to access TSO and CA Enterprise Tester.

Interactive System SESSION Command

Options of the Interactive System SESSION command or the Terminal Panel may be used to configure virtual terminals that have different features than the terminal that is used to access the CA Enterprise Tester Interactive System.

The SESSION command supports keywords that may be used to configure a virtual terminal.

- A pair of the keywords, 24X80, 32X80, 43X80or 27X132, may be specified to choose virtual terminal default and alternate screen sizes.
- One of the keywords, BASICDS or EXTDS, may be specified to configure a virtual terminal to support either the basic or extended 3270 data stream. The extended data stream allows an online system to explicitly state field and character color and to choose a variety of extended field and character highlighting options.
- A MODEL keyword and operand may be used to identify a VTAM Logon Mode Table entry, or LOGMODE name, that is to be used to start a session.

Virtual terminal features may also be configured using the Interactive System Terminal Panel. The Terminal Panel is displayed when the TERMINAL command is entered at the Control Panel or System Menu. The Terminal Panel is illustrated in the following example.

CA Enterprise Tester Terminal Panel

```
+-------------------------------------------------------------------------------+
| ProTerm Terminal Panel ------------------------------------------------------ |
| COMMAND ===>                                                                  |
|                                                                               |
| ENTER TO UPDATE                                                               |
| PF15 END PANEL DISPLAY                                                        |
|                                                                               |
| Specify virtual terminal options ( Defaults to TSO terminal values ):         |
|                                                                               |
| Default screen size ===> 24X80 ( 24X80, 32X80, 43X80 or 27X132 )              |
|                                                                               |
| Alternate screen size ===> 27X132 ( 24X80, 32X80, 43X80 or 27X132 )           |
|                                                                               |
| Terminal type ===> EXTDS ( BASICDS or EXTDS )                                 |
|                                                                               |
| Terminal model ===> MOD5 ( MOD2, MOD3, MOD4, MOD5 or logmode )                |
|                                                                               |
| Specify TSO terminal read options:                                            |
|                                                                               |
| Read Modified All? ===> YES ( Yes reads PA1, PA3, and PEN fields )            |
|                                                                               |
| Read Buffer? ===> NO ( Yes reads unformatted screen buffer )                  |
|                                                                               |
| Disable TSO Timeout? ===> NO ( Yes disables TSO S522 abend )                  |
|                                                                               |
+-------------------------------------------------------------------------------+
```

The CA Enterprise Tester Terminal Panel is displayed when a TERMINAL command, abbreviated TE, is entered at other CA Enterprise Tester panels. The Terminal Panel provides options which support the configuration of CA Enterprise Tester virtual terminals.

The name of the panel is displayed on the left of the first line and the upper right corner is used for standard ISPF short message display.

A standard ISPF command entry area on the second line of the panel permits entry of CA Enterprise Tester commands as well as ISPF commands such as SPLIT and SWAP, and execution of TSO commands.

The third line of the panel is the standard ISPF long message display area. A long message is displayed when you press the PF key to which you have assigned the HELP command, (usually PF1), when a short message is displayed, further describing an error or unusual condition, or giving other additional information. Not all short messages will have additional long messages.

Note: Panel formats are controlled by ISPF terminal and user options, so if your panels do not look like the examples, you can review your ISPF parameters by selecting Option 0 from the ISPF Primary Option Menu.

Virtual terminal options specified in a SESSION command override options specified on the Terminal Panel. Options specified on the Terminal Panel are remembered and override default options that are chosen to be compatible with the terminal being used to access CA Enterprise Tester.

Default Screen Size Option

One of the keywords, 24X80, 32X80, 43X80or 27X132, may be specified to choose the virtual terminal default screen size.

Alternate Screen Size Option

One of the keywords, 24X80, 32X80, 43X80or 27X132, may be specified to choose the virtual terminal alternate screen size.

Terminal Type Option

One of the keywords, BASICDS or EXTDS, may be specified to configure a virtual terminal to support either the basic or extended 3270 data stream. The extended data stream allows an online system to explicitly state field and character color and to choose a variety of extended field and character highlighting options.

Terminal Model Option

One of the keywords, MOD2, MOD3, MOD4, MOD5, or a logmode name may be used to identify a VTAM Logon Mode Table entry to be used to start a session.

Read Modified All Option

If YES is specified, upon input of PA1, PA3 or a light pen, modified fields are to be read from the terminal screen. If the CA Enterprise Tester Record Facility is active on the port, PA key or PEN input will be recorded if fields were modified prior to pressing a PA key or prior to selecting a PEN attention field.

Read Buffer Option

If YES is specified for the option, when a transaction is input at an unformatted screen, (one with no attributes), then the terminal buffer is to be read. The result is that unformatted screen input is stored in proper position in the virtual terminal screen, allowing applications such as the IMS/MFS screen painter to work correctly using a CA Enterprise Tester session in Zoom Mode.

Disable TSO Timeout Option

If you specify YES for this option, TSO S522 timeout abends will be disabled for TSO sessions using the virtual terminal. By turning off TSO timeouts, the TSO address space will not be cancelled when a very long running transaction is entered on a CA Enterprise Tester session. A system installation option controls how the terminal panel option works.

Depending on how the system option was specified, you may permanently disable this type of abend for your virtual terminal sessions, or you may disable the abend only for the life of the CA Enterprise Tester dialog.

Terminal Panel Commands

CA Enterprise Tester processing available at the Terminal Panel includes the following:

- If the Enter key is pressed, any changes to the virtual terminal configuration will be processed.
- While viewing the Terminal Panel, the END command may be entered to avoid updating the virtual terminal configuration and to return to the Control Panel or any other CA Enterprise Tester panel from which the TERMINAL command was invoked.

VTAM Logon Mode Table

All real and virtual terminals are associated with a VTAM Logon Mode Table. Logon mode tables are defined during installation by the system programmers who define devices to a network. A mode table contains entries that define SNA session protocols and may define screen sizes and other terminal features. Each entry in a mode table defines a different set of session protocols and terminal features.

When a session is started with an online system, using either a real or a virtual terminal, an appropriate entry in the mode table associated with the terminal is selected and the contents of the entry are sent to an online system with the session request. The information informs the online system of the capabilities of a device and may instruct the online system to query the device after a session is started, to obtain more detailed information.

Using the SESSION command MODEL keyword or the Terminal Panel Terminal Model field, CA Enterprise Tester allows a terminal user to name the specific mode table entry or to use one of the keywords MOD2, MOD3, MOD4or MOD5 in combination with either the BASICDS or EXTDS keyword to refer symbolically to mode table entries that were defined when CA Enterprise Tester was installed.

The following table lists Terminal Model keywords and their associated default and alternate virtual terminal screen sizes.

Model Keyword	Default Size	Alternate Size
MOD2	24 x 80	24 x 80
MOD3	24 x 80	32 x 80
MOD4	24 x 80	43 x 80
MOD5	24 x 80	27 x 132

When a mode table entry is not specified, CA Enterprise Tester selects an entry based on the default and alternate screen sizes and the 3270 data stream level selected for a session. Screen sizes and the 3270 data stream level may be specified using SESSION command keywords or Terminal Panel fields described or illustrated, and default to values supported by the terminal being used to access CA Enterprise Tester.

The information contained in the mode table entry that is used to start a session, is only a suggestion to an online system. When a session request is received by the online system, it may choose different screen sizes and other parameters based on terminal definitions maintained within the online system. For example, CICS allows terminal definitions to be created that fix the screen sizes used for sessions with a named terminal and to specify whether or not a device supports the extended data stream or is to be queried for additional device information after a session is started. When rigid online system definitions conflict with suggested values, values specified by the online system during session start up are used.

When a mode table entry is used to start a session, that does not contain screen sizes and that tells the online system that a query should be used to obtain device information, and when the online system honors this information and sends the suggested query, then CA Enterprise Tester responds automatically to the query using screen sizes and data stream level that are specified in a SESSION command, Option Table fields, or that match the terminal being used to access CA Enterprise Tester. If screen sizes are not specified, but one of the symbolic mode table entry names, MOD2, MOD3, MOD4 or MOD5 are specified, then CA Enterprise Tester responds to the query using the implied screen size combinations documented.

When a CA Enterprise Tester session is started by a CA Enterprise Tester script, a LOGON statement may specify a virtual terminal configuration using parameters that are equivalent to those supported by the SESSION command and Terminal Panel. If a LOGON statement does not specify a terminal configuration, then a default configuration, defined when CA Enterprise Tester was installed, is used. LOGON documents Script Language LOGON statement parameters.

Chapter 3: Interactive System Command References

This chapter documents the syntax of and options supported by all CA Enterprise Tester Interactive System commands, arranged alphabetically by command name.

- Interactive System Command Reference

Interactive System Command Reference

This section provides documentation of all Interactive System commands. The commands are presented in alphabetical order according to command name.

Syntax Notation Format

- CA Enterprise Tester Interactive System commands and keywords are in uppercase text
- User data is listed in lowercase text
- Optional parameters are enclosed in parentheses, ( )
- Alternative keywords and/or parameters are separated by the bar symbol, |

ANALYZE

The ANALYZE command, abbreviated A, displays the Analyze Panel.

Example:

```
COMMAND ===> a 1
```

Format:

```
ANALYZE ( port )
```

Port is optional and may be specified when the ANALYZE command is entered at the Control Panel. When entered at the Session Panel, the Analyze Panel is displayed for the port being used.

The Analyze Panel is displayed when a valid ANALYZE command is entered at the Control or Session Panel. A session analyzer may be used to automatically re-create previously traced CA Enterprise Tester sessions and to compare live session responses to previously traced online system responses.

Refer to Analyzing Traced Sessions and Regression Testing for complete documentation of CA Enterprise Tester session analyzers.

ATTN

The ATTN command, abbreviated AT, presses the Attention key on a CA Enterprise Tester virtual terminal keyboard.

Example:

```
COMMAND ===> at
```

Format:

```
ATTN
```

While the keyboard on a real terminal used to access CA Enterprise Tester is unlocked, the ATTN command may be used to send an Attention key signal to an online system being accessed using a CA Enterprise Tester session.

When an ISPF panel is displayed and the terminal keyboard is unlocked, ISPF is in control of real terminal input. If the Attention key (or PA1 key) is pressed while ISPF is in control of terminal input, ISPF either ignores the signal or terminates the ISPF application. The ATTN command provides a way to send an Attention key signal to an online system that is not intercepted and processed by ISPF.

While a CA Enterprise Tester session is displayed in Zoom Mode, CA Enterprise Tester is in control of terminal input. If the Attention key is pressed while the terminal keyboard is unlocked CA Enterprise Tester sends the Attention key signal through to the online system.

While the terminal keyboard is locked, CA Enterprise Tester is in control of the Attention key. If the Attention key is pressed while the VIEW command is active or while a STEP or REVIEW command that specified multiple session steps is active, and CA Enterprise Tester is not waiting on a response from an online system, then the active command is stopped. If CA Enterprise Tester is waiting on a script, the script is stopped.

If the Attention key is pressed while CA Enterprise Tester is waiting for an online system response, the CA Enterprise Tester Session Wait Panel is displayed. The Session Wait Panel allows the user to specify an action. The ATTN command may be entered to send an Attention key signal to the online system. The INVITE command may be entered to resume the wait for the online system response. The END command may be entered to end the wait for the online system response and to return to a higher level panel. When a session wait is ended, the session is timed out. While a session is timed out, an END SESSION command may be entered to terminate the session, or the session may be selected to resume the wait for the online system response.

BROWSE

The BROWSE command, abbreviated B, displays the Browse Panel. The Browse Panel is used to start a display of a trace list or trace data set on a port without creating another online session.

Example:

```
COMMAND ===> b 1
```

Format:

```
BROWSE ( port )
```

port is optional and may specify a port number on which to initiate the trace browse function. If a port is not entered, CA Enterprise Tester will default to the next available port.

The Browse Panel is displayed when a BROWSE command is entered at the Control Panel. A trace browse may be used to automatically create a movie of previously traced sessions without creating a live online system session. Previously traced sessions may be reproduced for demonstration purposes without affecting the operation of the online system or the data accessed by the system. Either a trace data set or a trace list may be used as input to the browse.

Refer to Browsing a Traced Session for complete documentation of the CA Enterprise Tester Browse Facility.

CLEAR

The CLEAR command, abbreviated C, presses the Clear key on a CA Enterprise Tester virtual terminal keyboard.

Example:

```
COMMAND ===> c
```

Format:

```
CLEAR
```

Additional Information:

The real Clear key on a terminal keyboard is not supported by ISPF. The CLEAR command allows the virtual terminal Clear key to be pressed while using a CA Enterprise Tester Session in Window Mode.

While using a CA Enterprise Tester Session in Zoom Mode, the real Clear key or a PF key that has been assigned to the CLEAR command while Zoom Mode is active, can be used to press a virtual terminal Clear key.

CODE

The CODE command, abbreviated CO, displays a Security Panel.

Example:

```
COMMAND ===> co
```

Format:

```
CODE
```

CA Enterprise Tester provides password security features which permit the capture and use of procedures used to sign on to online systems, including data which represents secure passwords, and ensure that actual passwords remain confidential.

When the Interactive System CODE command is entered, panels and options displayed will depend on the active password security option and other installation options specified by the systems programmer who installed CA Enterprise Tester.

For complete documentation, please refer to Password Security.

DOWN

The DOWN command displays subsequent data in the scrollable areas of a CA Enterprise Tester Control Panel and Monitor Panel, and panels which display virtual terminal screen images.

Example:

```
COMMAND ===> down max
```

Format:

```
DOWN ( number | PAGE |  DATA | HALF |  CSR | MAX )
```

Number specifies the number of lines to scroll;

- PAGE will scroll one page of data;
- DATA will scroll one page of data, minus one line;
- HALF will scroll one-half page of data;
- CSR will scroll to the position of the cursor within the data;
- MAX will scroll to the bottom margin of the data.

If an amount is not specified, the scroll amount is taken from the panel's SCROLL amount field.

The UP and DOWN commands may be used to scroll through the list of ports displayed on the CA Enterprise Tester Control Panel.

The UP, DOWN, LEFT and RIGHT commands may be used to scroll within a virtual terminal screen image when the image is larger than the terminal screen being used to access CA Enterprise Tester.

The UP, DOWN, LEFT and RIGHT commands may be used to scroll through the Monitor Panel's Activity Log.

END

The END command, abbreviated E, ends the display of a panel and returns to the next higher level panel, or terminates one or more sessions, scripts, recordings, traces and other functions.

Example:

```
COMMAND ===> e t
```

Format:

```
END ( options )
```

Options specifies what is to be ended and may be one of the functions: SESSION, PLAY, ANALYZE, BROWSE, RECORD, TRACE, LOG, and/or a port number.

Additional Information:

At most CA Enterprise Tester panels, if a function is not specified, the END command ends the display of the current panel and displays the next higher level panel.

At a Session Panel or a Monitor Panel, you may end a session, script, recording or trace that is active on the port being used or end an Interactive System Log by entering one of the following commands:

- END SESSION; which may be abbreviated, (e s)
- END PLAY; (e p)
- END ANALYZE; (e a)
- END RECORD; (e r)
- END TRACE; (e t) or
- END LOG; (e log)

When entered at the Control Panel, the END command may specify:

- A function to be ended;
- A port on which all functions are to be ended;
- A function and a port, in order to end a single function on a specified port.

When the END command is entered at the Log Panel, any active log data set is closed and data set logging is ended.

If the Attention key is pressed while the terminal keyboard is locked and while waiting on a required response from an online system, then a Session Wait Panel is displayed. The END command may be entered to end the wait for the online system response and to time the session out. A timed-out session may be selected at a later time to resume the wait.

ERASE

The ERASE command, abbreviated ER, presses the Erase EOF key on a CA Enterprise Tester virtual terminal keyboard.

Example:

```
COMMAND ===> er
```

Format:

```
ERASE
```

While using a CA Enterprise Tester Session in Window Mode or split-screen mode, or if a virtual terminal screen is larger than the real terminal screen being used to access CA Enterprise Tester, then the full virtual terminal screen can not be displayed all at once. If a field is erased using a terminal's real Erase EOF key, then only the visible portion of the erased field will be cleared to nulls.

The ERASE command clears the remainder of the virtual terminal screen field at which the cursor is positioned, even if the field is not fully displayed.

EXECUTE

The EXECUTE command, abbreviated EX, executes one or more Script Language statements on one or more ports on which a played script is active.

Example:

```
COMMAND ===> ex 'think 500'
```

Format:

```
EXECUTE ( options )
```

Options may include one or more Script Language statements enclosed in single or double quotes and a port number if entered at the Control Panel.

If no options are specified, the EXECUTE command will recall the last EXECUTE command entered.

The EXECUTE command may be entered while using a CA Enterprise Tester Session Panel or a Monitor Panel on a port on which a played script is active to execute the Script Language statements on the current port.

When entered at the Control Panel, an EXECUTE command may specify the port on which specified Script Language statements are to be executed. If a port is not specified, the statements are executed on all ports on which a played script is active.

If a script is actively executing on a port when the EXECUTE command is entered, then the script will be temporarily stopped after the script statement currently being executed has completed. The Script Language statements specified by the EXECUTE command will then be executed, after which script execution is automatically resumed.

See Script Monitoring and Interactive Debug, Advanced Analysis Techniques, and Multiple Session Testing for examples of the uses of the EXECUTE command.

HELP

The HELP command displays long error messages or the CA Enterprise Tester Help panels.

Example:

```
COMMAND ===> help
```

Format:

```
HELP
```

CA Enterprise Tester provides a set of commands to navigate through the CA Enterprise Tester Tutorial. Standard ISPF HELP commands such as BACK, UP or TOC may be also be used.

When an error message is displayed in the upper right corner of a CA Enterprise Tester panel, the HELP key or command will display a more detailed description of the message on line 3 of the panel. If no message is displayed or if the HELP key is pressed again, the CA Enterprise Tester Help panels are displayed using standard ISPF Tutorial facilities.

INVITE

The INVITE command, abbreviated I, displays messages sent by an online system after the online system has previously sent a message that freed the terminal keyboard. The INVITE command may also used to control automatic INVITE processing.

Example:

```
COMMAND ===> i
```

Format:

```
INVITE ( options )
```

Options may specify a time interval to be used during subsequent automatic INVITE processing, or may be the question mark character, ?, to request the display of the current automatic INVITE interval.

While using a CA Enterprise Tester session, if the INVITE command is entered and no option is specified, CA Enterprise Tester checks to see if an online system being accessed via a CA Enterprise Tester Session has sent a message since the terminal keyboard was last freed, and receives and displays any message sent. The INVITE command may be used while using a CA Enterprise Tester session in Window Mode to receive delayed messages that could not be displayed automatically due to restrictions in ISPF terminal display services. The INVITE command should not be required when using a session in Zoom Mode.

If the Attention key is pressed while using a CA Enterprise Tester session and while the keyboard is locked while waiting on a required response from an online system, then the CA Enterprise Tester Session Wait Panel is displayed. The INVITE command may be entered at that Session Wait Panel to resume the wait for the online system response.

When INVITE ? is entered, the current automatic INVITE time interval is displayed on the command line.

When a time interval of the form ssth expressing seconds (ss), and tenths and hundredths of seconds (th), is entered as the operand of an INVITE command, then the automatic INVITE time interval is reset to the specified value. If the INVITE interval is set to zero (0), then automatic INVITE processing is turned off.

The INVITE interval may be set to a non-zero time to activate automatic INVITE processing. When an online system sends a message that frees the terminal keyboard (and ends the session bracket), then CA Enterprise Tester will display the message received, leaving the terminal keyboard locked, and will wait for up to the specified interval. If another message is received within the interval, the message is displayed, again leaving the keyboard locked, and the wait is repeated until no additional message is received within an interval, at which time the terminal keyboard is freed.

The automatic INVITE interval may also be viewed and controlled using the CA Enterprise Tester Options Panel. The Options Panel is displayed when the OPTIONS command, abbreviated o, is entered at other CA Enterprise Tester panels.

Refer to See Using CA Enterprise Tester Sessions for an in-depth description of the purpose and use of the INVITE command.

JUMP

The JUMP command, abbreviated J, is used to specify the action to be taken by CA Enterprise Tester, when an attempt is made to exit CA Enterprise Tester using the ISPF JUMP(=), RETURN, or END command, while CA Enterprise Tester Sessions or played scripts are active.

Example:

```
COMMAND ===> jump off
```

Format:

```
JUMP ON | OFF
```

Additional Information:

While JUMP is OFF, CA Enterprise Tester can be exited only when no CA Enterprise Tester Session or played script is active.

While JUMP is ON, if the user enters an ISPF JUMP, RETURN or END command while a CA Enterprise Tester session or played script is active, CA Enterprise Tester will automatically terminate all active sessions and scripts before returning to ISPF.

JUMP command control may also be viewed and altered via the CA Enterprise Tester Options Panel. The Options Panel is displayed when the OPTIONS command, abbreviated o, is entered via other CA Enterprise Tester panels.

KEYS

The KEYS command, abbreviated K, displays the CA Enterprise Tester Keys Panel.

Example:

```
COMMAND ===> k
```

Format:

```
KEYS
```

The Keys Panel displays all current PF key assignments and a list of all commands supported by CA Enterprise Tester. The Keys Panel may be used to assign CA Enterprise Tester commands to PF keys.

The Keys Panel is fully documented in PF Keys and Commands.

LEFT

The LEFT command displays data that is to the left of the current display, in scrollable areas of the CA Enterprise Tester Monitor Panel, and panels which display virtual terminal screen images.

Example:

```
COMMAND ===> left max
```

Format:

```
LEFT ( number | PAGE |  DATA | HALF |  CSR | MAX )
```

- Number specifies the number of columns to scroll;
- PAGE will scroll one page of data;
- DATA will scroll one page of data, minus one column;
- HALF will scroll one-half page of data;
- CSR will scroll to the position of the cursor within the data;
- MAX will scroll to the left margin of the data.

If an amount is not specified, the scroll amount is taken from the panel's SCROLL amount field.

The UP, DOWN, LEFT and RIGHT commands may be used to scroll within a virtual terminal screen image when the image is larger than the terminal screen being used to access CA Enterprise Tester.

The UP, DOWN, LEFT and RIGHT commands may be used to scroll through the CA Enterprise Tester Monitor Panel's Activity Log.

LOCATE

The LOCATE command, abbreviated L, displays specified scrollable data on the CA Enterprise Tester Control Panel or CA Enterprise Tester Monitor Panel, or a specified screen from the CA Enterprise Tester Browse Panel.

Example:

```
COMMAND ===> l 10
```

Format:

```
LOCATE port-number |line-number |screen-number
```

- port-number specifies a port to display
- line-number specifies a line to display
- screen-number specifies a screen to display

The LOCATE command may be used to:

- Display a specified port on the CA Enterprise Tester Control Panel;
- Display a specified line of data in the CA Enterprise Tester Monitor Panel's Activity Log;
- Position a browse display at a specific screen image.

LOG

The LOG command displays the CA Enterprise Tester Log Panel.

Example:

```
COMMAND ===> log
```

Format:

```
LOG
```

The LOG command may be entered at most CA Enterprise Tester panels to display the Log Panel. (LOG has no abbreviation. The LOCATE command is abbreviated L, and LO would be an ambiguous abbreviation.) The Log Panel is used to open a log data set to capture script logs and other data and messages.

A script log is the log written to a PTELOG data set when a script is executed as a batch job, or the Message Log that is displayed on the Monitor Panel when a script is played interactively. Unlike session traces and recordings that are started on a port, the log that is started using the LOG command and panel is a central log. The log captures messages that are logged on all active ports in a single data set.

CA Enterprise Tester provides a utility menu option which will sort the merged log into port order, allowing all messages for a specific port to be viewed separately from all messages logged by other concurrently executed scripts.

Refer to Script Logs and Diagnostic Traces, for a description of the Interactive System Log feature and options supported by the Log Panel.

You may also refer to CA Enterprise Tester Utilities for a description of CA Enterprise Tester utilities.

MONITOR

The MONITOR command, abbreviated M, displays the CA Enterprise Tester Monitor Panel.

Example:

```
COMMAND ===> m 1
```

Format:

```
MONITOR ( options )
```

Options may specify a port number at the Control Panel, or may specify the script to be played and other parameters when entered at the Play Panel.

The MONITOR command may be entered at the Control Panel to monitor the execution of a script being played on a specified port or to monitor a script executed using the CA Enterprise Tester Analyze Panel.

While a script is active on a port, the MONITOR command may be entered at the Session Panel to monitor the script. The MONITOR command may be used to view the output or status of a script executed using either the Analyze Panel or the Play Panel.

The MONITOR command may be entered from the Play Panel to request that a script be loaded for execution and that the Monitor Panel be displayed prior to beginning script execution. Data set name, member name and script parameters may be entered as free-form options of the MONITOR command, or options may be entered via Play Panel fields.

Refer to Script Monitoring and Interactive Debug for complete documentation of the Monitor Panel.

OPTIONS

The OPTIONS command, abbreviated O, displays the CA Enterprise Tester Options Panel. The Options Panel displays current CA Enterprise Tester option settings and allows CA Enterprise Tester options to be changed.

Example:

```
COMMAND ===> o
```

Format:

```
OPTIONS
```

The OPTIONS command may be entered at any time while using a Control Panel, a Session Panel, a Monitor Panel or other CA Enterprise Tester panels.

CA Enterprise Tester Options and the Options Panel are fully documented in Options and Controls.

PA1-PA3

The PA1, PA2 and PA3 commands press the named keys on a CA Enterprise Tester virtual terminal keyboard.

Example:

```
COMMAND ===> pa1
```

Format:

```
keyname
( PA1 | PA2 | PA3 )
```

The real PA1, PA2and PA3 keys on a terminal keyboard are either unsupported or are assigned to special functions by TSO and ISPF. The PA1, PA2 and PA3 commands allow the virtual terminal's PA keys to be pressed by entering the PA1, PA2 and PA3 commands in the Session Panel command field, displayed in Window Mode, or by pressing a PF key that has been assigned to a PAn command.

While using a CA Enterprise Tester Session in Zoom Mode, the real PA1 and PA3 keys are not assigned to functions by TSO, and operate as virtual terminal keyboard keys. The PA2 key is assigned to the TSO RESHOW function. CA Enterprise Tester provides an option in the Options Panel section which controls the way a PA2 key is processed by the Interactive System when session displays are presented in Zoom Mode.

Should RESHOW (PA2) switch to Window Mode display?

If the option is set to YES, CA Enterprise Tester re-displays the current session in Window Mode when the PA2 key is pressed. If the option is set to NO, CA Enterprise Tester will stay in Zoom Mode, and entry of the PA2 key will invoke the TSO RESHOW function.

A CA Enterprise Tester Terminal Panel option, Read Modified ALL? allows a user to specify that upon input of a PA1, PA3or PEN command or key, modified fields are to be read from the screen. The result is that PA/PEN input is recorded if fields were modified prior to pressing a PA key or prior to selecting a PEN attention field.

CA Enterprise Tester Options and the Options Panel are fully documented in Options and Controls.

The CA Enterprise Tester Terminal Panel is documented in Configuring Virtual Terminals.

PEN

The PEN command simulates light pen or cursor select operations on the field attribute and designator character of the pen detectable field at which the cursor is positioned in a CA Enterprise Tester virtual terminal screen image.

Example:

```
COMMAND ===> pen
```

Format:

```
PEN
```

While ISPF does not provide general support for pen detectable fields, CA Enterprise Tester provides the PEN command that may be assigned to a Window Mode PF key or may be entered in the Command field while using a session via the CA Enterprise Tester Session Panel.

When the cursor is in a pen selection field, and the field is currently deselected, the PEN command sets the field MDT and changes the question mark, ? , designator character to a greater than character, > , indicating that the field is selected. If the field is currently selected, the PEN command resets the MDT and changes the > designator character to ? indicating that the field is deselected.

When a PEN command is entered with the cursor positioned in a null pen attention field, the field MDT is set and a pen aid and pen datastream is sent to the connected online system.

When a PEN command is entered with the cursor positioned in an & pen attention field, the field MDT is set and an Enter key AID and datastream is sent to the connected online system.

The PEN command may assigned to both Window Mode and Zoom Mode PF keys.

When using applications that issue a terminal read upon receipt of a pen or PA key AID, correct operation of light pen, cursor select key or PA key functions may require that the user activate the Read-Modified-All option on the CA Enterprise Tester Terminal Panel.

The CA Enterprise Tester Terminal Panel is documented in Configuring Virtual Terminals.

PF1-PF24

The PF1 through PF24commands press the named keys on a CA Enterprise Tester virtual terminal keyboard.

Example:

```
COMMAND ===> pf3
```

Format:

```
keyname
```

keyname is the name of a PF key, PF1 through PF24.

Under ISPF, and while using the CA Enterprise Tester Interactive System, all real PF keys are assigned to commands. CA Enterprise Tester provides PF1 through PF24 commands that may be entered via the Session Panel's command field, displayed in Window Mode, to press the named PF key on a virtual terminal keyboard. The CA Enterprise Tester Keys Panel allows terminal PF keys to be assigned to PFn commands, making it possible to use real PF keys to press a virtual terminal's PF keys.

Refer to PF Keys and Commands for a complete description of CA Enterprise Tester PF key support.

PLAY

The PLAY command, abbreviated P, displays the CA Enterprise Tester Play Panel or initiates the execution of a played script.

Example:

```
COMMAND ===> p 1
```

Format:

```
PLAY ( options )
```

Options may specify a port number when entered at the Control Panel or may specify the script to be played and other parameters when entered at the Play Panel.

The PLAY command may be entered at:

- The Control Panel to display the Play Panel, in order to execute a CA Enterprise Tester script on a specified port;
- The Session Panel command field in Window Mode to play a script on the CA Enterprise Tester port currently being used; the PLAY command then displays the Play Panel;
- The Play Panel to request that a script be played asynchronously in the TSO background and that the CA Enterprise Tester Control Panel be displayed while the script is executing.

Data set name, member name and script parameters may be entered as free-form options of the PLAY command, or options may be entered via Play Panel fields.

By using ISPF command stacking, the free-form data entry capability allows a script to be played directly from the Control or Session Panels, skipping the actual display of the Play Panel.

For example, the following stacked commands may be entered at the Control Panel to play a named script, on port 1, and in the TSO background:

```
COMMAND ===> p 1;p 'script.data.set.name' script1
```

Refer to Playing Session Scripts for complete documentation of the CA Enterprise Tester Play Panel.

RECORD

The RECORD command, abbreviated R, displays the CA Enterprise Tester Record Panel. The Record Panel is used to start a recording of session activity on a port. While a session recording is active, the RECORD command may be used to add comments or other Script Language statements to the session recording.

Example:

```
COMMAND ===> r 1
```

Format:

```
RECORD ( options )
```

Options may specify a port number when the RECORD command is entered at the CA Enterprise Tester Control Panel. When entered at the Session or Monitor Panel on a port on which a recording is active, the RECORD command may specify literal data to be written to the active recording.

The RECORD command may be entered at:

- The Control Panel to display the Record Panel, in order to start a recording of subsequent session activity on a specified port;
- The Session Panel or Monitor Panel to start a recording of session activity on the current port. The RECORD command then displays the Record Panel.

- The Session Panel or Monitor Panel to add a comment statement to a script.

A session recording captures session events in the form of executable Script Language statements, that when later executed, re-create the recorded session activity. A recording will capture session events resulting from manual use of a CA Enterprise Tester session and/or session events that are created by a script that is executed on the port on which a recording is active.

When a RECORD command is entered at the Session Panel or Monitor Panel on a port on which a recording is active, the command may specify a string of data, enclosed in single or double quotes, that is to be immediately added to the session recording.

For example, the following command adds a Script Language comment statement to the recording:

```
COMMAND ===> r '* this comment is added to the session recording'
```

An active recording may be temporarily suspended and may be continued at a later time by entering the STOP RECORD and RESUME RECORD commands. A recording may be terminated at any time by entering the END RECORD command.

Refer to Recording Session Scripts, for a complete documentation of the CA Enterprise Tester session recording feature.

REPLY

The REPLY command, abbreviated REP, is used to reply to a Script Language WTOR that has been issued by a CA Enterprise Tester script that is executed under the control of the CA Enterprise Tester Interactive System.

Example:

```
COMMAND ===> rep 'retry'
```

Format:

```
REPLY options
```

Options must include a literal value contained in single or double quotes and may specify a port number when entered at the Control Panel.

The REPLY command may be entered while using the CA Enterprise Tester Session or Monitor Panels on a port on which a played script is waiting for a response to a previously executed WTOR statement.

In addition to the literal reply, when entered from the CA Enterprise Tester Control Panel, a REPLY command may also specify the port to which the reply is to be directed. If a port is not specified, a reply will be directed to each outstanding WTOR on all ports.

A script that was waiting on a reply is resumed when the REPLY command is entered.

Refer to WTOR for a description of the Script Language WTOR statement.

RESUME

The RESUME command, abbreviated RES, resumes a stopped script, recording or trace on the port currently being used or on a specified port.

Example:

```
COMMAND ===> res
```

Format:

```
RESUME ( options )
```

Options may include a function, PLAY, ANALYZE, RECORD or TRACE; or when entered at the Control Panel, the RESUME command may also or alternatively specify a port number.

The RESUME command may be entered at:

- The Control Panel to resume a stopped play, recording or trace on one or all CA Enterprise Tester ports. If a function is not specified, all stopped functions on a specified port are resumed. If a port is not specified, a named function is resumed on all ports on which the function is active, but stopped. If no options are specified, all stopped functions on all ports are resumed.
- The Session Panel or Monitor Panel to resume a stopped script, recording or trace on the current port. If a function is not specified, all stopped functions on the port are resumed.

When a stopped recording is resumed, subsequent session activity on the port will be recorded.

When a stopped trace is resumed, subsequent session activity on the port will be traced.

When a stopped script is resumed, including scripts started using the PLAY or ANALYZE command, the script will resume execution in the TSO background, freeing the terminal keyboard to perform other activity while the script is executing.

RETURN

The RETURN command conditionally causes CA Enterprise Tester to terminate all active functions and to return to ISPF.

Example:

```
COMMAND ===> return
```

Format:

```
RETURN
```

The action of the RETURN command is affected by the current setting of the JUMP option specified on the CA Enterprise Tester Options Panel. The Options Panel is displayed when the OPTIONS command is entered at other CA Enterprise Tester panels.

When the JUMP option is ON, the RETURN command causes CA Enterprise Tester to automatically end all active sessions, scripts, recordings and traces prior to returning to ISPF.

When the JUMP option is OFF, the RETURN command will return to ISPF only when sessions and scripts are not active. When sessions or scripts are active, and when JUMP is OFF, the RETURN command will display the status of active CA Enterprise Tester functions via the Control Panel.

Sessions and scripts may be ended using END SESSION, END PLAY or END ANALYZE commands, after which, the END or RETURN commands may be entered to return to ISPF.

REVIEW

The REVIEW command, abbreviated REV, displays prior session or trace screen images.

Example:

```
COMMAND ===> rev
```

Format:

```
REVIEW ( options )
```

Options may specify which screens are to be reviewed, either session, S , or trace T , and/ or a number of screen images to display.

Additional Information:

Screen images, reflecting recent session activity on a port, are saved for subsequent review if a non-zero review limit has been specified using the CA Enterprise Tester Options Panel.

If a script or session analyzer is executed on a port that uses the Script Language TRACK statement to read screen images from a previously created session trace data set, then a second set of screen images most recently read from the data set is saved for review. The number of trace screen images saved is also subject to the review limit option setting.

The REVIEW command may be entered at the Session Panel command field while using a session interactively in Window Mode, while reviewing session screens, or may be entered at the Monitor Panel, to display a recent session or trace screen. A REVIEW command may specify either Session or Trace to select which set of review screens are to be displayed.

A REVIEW command may specify a number to back up more than one review screen at a time. When a number is specified, and a step display time interval is specified on the Options Panel, the REVIEW command displays each progressively older screen for the specified time interval before automatically displaying the next older screen.

If the error message, No Review screens is displayed as the result of a failed REVIEW command, the Maximum number of session screens to save for REVIEW option on the CA Enterprise Tester Options Panel should be examined and changed to an appropriate number to support the review activity.

Refer to Interactive Session Review for complete documentation of the CA Enterprise Tester Review feature.

RIGHT

The RIGHT command displays data that is to the right of the current display, in scrollable areas of the CA Enterprise Tester Monitor Panel, and panels which display virtual terminal screen images.

Example:

```
COMMAND ===> right max
```

Format:

```
RIGHT ( number | PAGE |DATA   | HALF |CSR | MAX )
```

- Number specifies the number of columns to scroll;
- PAGE will scroll one page of data;
- DATA will scroll one page of data, minus one column;
- HALF will scroll one-half page of data;
- CSR will scroll to the position of the cursor within the data;
- MAX will scroll to the right margin of the data.

Additional Information:

If an amount is not specified, the scroll amount is taken from the panel's SCROLL amount field.

The UP, DOWN, LEFT and RIGHT commands may be used to scroll within a virtual terminal screen image when the image is larger than the terminal screen being used to access CA Enterprise Tester.

The UP, DOWN, LEFT and RIGHT commands may be used to scroll through the CA Enterprise Tester Monitor Panel's Activity Log.

ROLL

The ROLL command, abbreviated RO, has two functions: While using the Monitor Panel or while using an active session, the ROLL command switches to the Monitor Panel or active session on the next port on which a script or session is active; while reviewing screen images, the ROLL command toggles to the corresponding session or trace review screen on the current port.

Example:

```
COMMAND ===> ro
```

Format:

```
ROLL
```

When entered at the Monitor Panel or while using an active session, the ROLL command functions only when scripts or sessions are active on other ports.

When entered while reviewing screen images, the ROLL command functions only when both session and trace review screens are available on the current port.

The Maximum number of session screens to save for REVIEW option on the CA Enterprise Tester Options Panel controls the number of session screens which will be available for review. This number should be examined and changed if the error message, No session screen is displayed as the result of a failed ROLL command.

SECURE

The SECURE command, abbreviated SEC, displays a CA Enterprise Tester Security Panel.

Example:

```
COMMAND ===> sec
```

Format:

```
SECURE
```

CA Enterprise Tester provides password security features which permit the capture and use of procedures used to sign on to online systems, including data which represents secure passwords, and ensure that actual passwords remain confidential.

When the Interactive System SECURE command is entered, panels and options displayed will depend on the active password security option and other installation options specified by the systems programmer who installed CA Enterprise Tester.

For complete documentation, please refer to Password Security.

SELECT

The SELECT command, abbreviated S, is a variation of the SESSION command which may be entered at CA Enterprise Tester panels, such as the Control Panel or the Monitor Panel to select an active session for interactive use.

Refer to SESSION for complete documentation of the SESSION command.

SESSION

The SESSION command, abbreviated S, is used to start a new session, select an active session, or to display a System Menu that may be used to start a session.

Examples:

```
COMMAND ===> s cics1
```

```
COMMAND ===> s 2
```

Format:

```
SESSION ( number )
( system-name )
( FROM terminal-name )
( MODEL terminal-model
  MOD2 | MOD3 |
  MOD4 | MOD5 )
( default-screen-size
  24X80 | 32X80 |
  43X80 | 27X132 )
( alternate-screen-size
  24X80 | 32X80 |
  43X80 | 27X132 )
( terminal-type BASICDS | EXTDS )
( logon-data )
```

- Port specifies the number of the port on which a session is to be started or selected; if a number is not specified, then a session is started on the first port on which no session or played script is active. If scripts or sessions are active on one or more ports, then a port number may be required to make your intentions clear.
- System-name is the one through eight-character name of an online system with which a session is to be started;
- FROM is an optional keyword and indicates that the data item that follows is a virtual terminal name;
- Terminal-name is a data item identified by the FROM keyword, and is the name of the virtual terminal to be used for the session. terminal-name is the actual name of a VTAM APPL that has been defined at an installation for use as a CA Enterprise Tester virtual terminal; if the FROM keyword is not specified; then a virtual terminal will be assigned automatically;
- MODEL is an optional keyword and indicates that the data which follows is a symbolic terminal model name or the name of a mode table entry;
- Terminal-model is a data item identified by the MODEL keyword, and is the logon mode table entry to be used to start a session. Terminal-model may be one of the keywords MOD2, MOD3, MOD4 or MOD5, to refer symbolically to actual mode table entry names defined at an installation, or may be the actual name of a logon mode table entry. The mode table and mode table entries associated with CA Enterprise Tester virtual terminals are defined when CA Enterprise Tester is installed.
- Default-screen-size is optional, and is the default virtual terminal screen size to be used for a session. default-screen-size may be one of the keywords 24X80, 32X80, 43X80 or 27X132;

- Alternate-screen-size is optional, and is the alternate virtual terminal screen size to be used for a session. alternate-screen-size may be one of the keywords 24X80, 32X80, 43X80 or 27X132;
- Terminal-type is optional, and specifies the type of terminal to be used for a session. terminal-type may be one of the keywords BASICDS or EXTDS;
- Logon-data is any other SESSION command data entered, and is free-form logon data that is to be sent to the online system with the session request.

To start a session, enter the SESSION command and the name of the online system with which you wish to initiate a session. Optionally, you may specify a port. If you do not specify a port, then by default, the session will be started on the next available port.

If a script is active on a port, it will not be selected by the SESSION command unless the port number is specified.

To select an active session, enter the SESSION command and the port number of the desired session.

If a SESSION command is entered when no sessions are active, or if a session is not active on a specified port and an online system name is not specified, then the CA Enterprise Tester System Menu is displayed. The System Menu is tailored by each installation and may provide a list of available system names and instructions that describe how to start a session.

When a SESSION command is entered at the Control Panel, then other optional session parameters may also be specified:

The SESSION command may also be entered at the Monitor Panel, or at the Session Panel while reviewing session screens or an active session step, to select the session that is active on the current port for interactive use.

Refer to Starting Virtual Terminal Sessions for additional information and examples that show how to start CA Enterprise Tester sessions.

The SELECT command and the START command are also provided as logical variations of the SESSION command.

SPOT

The SPOT command, abbreviated SP, displays the current cursor location within a virtual terminal screen image.

Example:

```
COMMAND ===> sp
```

Format:

```
SPOT
```

The SPOT command may be entered at a CA Enterprise Tester panel command field while viewing a virtual terminal screen image in Window Mode. When the SPOT command is entered, the current cursor location is displayed in the form <r,c>, where r is the cursor row and c is the cursor column. The row and column displayed is relative to the virtual terminal screen. The SPOT command may be used to find screen locations for use in user-written CA Enterprise Tester scripts. After positioning the cursor in a Session or Session Step Panel to use the SPOT command, the cursor will remain in the current position. In a Review Panel, however, pressing the Enter key following the SPOT command execution will return the cursor to the original position within the panel.

START

The START command, abbreviated S, is a variation of the SESSION command which may be entered at the Control Panel to start a session with an online system.

Refer to SESSION for complete documentation of the SESSION command.

STEP

The STEP command, abbreviated STE, may be used to advance displayed screen images one or more session steps; or it may be used to resume an active script on the current port and allow the script to drive a current session one or more session steps.

Example:

```
COMMAND ===> ste
```

Format:

```
STEP ( number )
```

Number may optionally be entered to advance a specific number of steps.

A session step is defined to be a point in a CA Enterprise Tester session at which a message is sent from a CA Enterprise Tester virtual terminal to an online system or at which a message from an online system is received at a virtual terminal.

Virtual terminal screen images of a current session are presented to an Interactive System user via a panel titled, Pro Term Session.

Virtual terminal screen images as a result of an analyzer or any other CA Enterprise Tester script are presented to an Interactive System user via a panel titled, Pro Term Session Step.

While viewing a screen image at a CA Enterprise Tester Session Step panel, from a current session driven by a script that is stopped, the STEP command may be entered to resume the script, driving the session for one more session steps.

When a message is sent or received on the session, the script is stopped at the session step, and the virtual terminal screen is then displayed at a CA Enterprise Tester Session Step Panel.

The STEP command can be also be entered at a CA Enterprise Tester Monitor Panel to resume a script, driving the session until one or more session steps have been completed.

A number may be specified to automatically advance a specified number of session steps.

The STEP command may be entered while reviewing session or traced screens, to advance to the next session step. If a number is specified, the review is advanced the specified number of session steps. When a number is specified that is larger than the remaining number of review screens, and a script is active on the port, then the specified number is reduced by one for each subsequent review screen, and the script is automatically resumed to drive the session until the specified number of steps have occurred.

When a STEP command specifies more than one session step, and a non-zero time interval has been specified for the step display time option provided on the CA Enterprise Tester Options Panel, then each session step is displayed for the specified interval, before automatically proceeding to the next session step. If a zero step display time interval has been specified, session steps are not displayed, and the STEP command proceeds directly to the specified step.

If a script encounters an error or other stop condition prior to reaching a session step, the situation is presented via the Monitor Panel.

STOP

The STOP command, abbreviated STO, stops one or more active CA Enterprise Tester Scripts, Traces or Recordings.

Example:

```
COMMAND ===> sto t
```

Format:

```
STOP ( options )
```

Options may include one of the functions, PLAY, ANALYZE, RECORD or TRACE. When entered at the Control Panel, a port number may be specified.

The STOP command may be entered at the Control Panel to stop the specified function(s) on the specified port(s).

- A port may be specified by placing the cursor on the port to be stopped.
- A function may be specified by placing the cursor on the function to be stopped.
- If both a function and a port are specified, the function is stopped on the specified port.
- If a port is not specified, a specified function is stopped on all ports on which it is active.

- If a function is not specified, all active functions on a specified port are stopped.
- If no options are specified, all active functions on all active ports are stopped.

The STOP command may be entered at a Monitor Panel or a Session Panel to stop the specified function on the current port. If a function is not specified, all active functions on the port are stopped.

When a script is stopped, the script will be interrupted when the Script Language statement currently being executed has completed. The VIEW, STEP, or RESUME commands can be entered at a later time to continue execution of the script.

When a trace is stopped, screen images reflecting session activity on the port are not captured. The trace data set remains open. The RESUME command can be entered at a later time to resume a stopped session trace.

When a recording is stopped, Script Language statements reflecting session activity on the port are not generated. The recording data set remains open. The RESUME command can be entered at a later time to resume a stopped session recording.

TERMINAL

The TERMINAL command, abbreviated TE, displays the CA Enterprise Tester Terminal Panel. The Terminal Panel may be used to define a default virtual terminal configuration to be used when CA Enterprise Tester sessions are subsequently started using the SESSION command or CA Enterprise Tester System Menu.

Example:

```
COMMAND ===> te
```

Format:

```
TERMINAL
```

By default, when the SESSION command or System Menu is used to start a session, a virtual terminal is configured using screen sizes that match the terminal being used to access TSO, ISPF and CA Enterprise Tester, and to support the 3270 data stream level that is supported by the terminal. Values specified on the Terminal Panel override default values. Values specified in a SESSION command override values specified on the Terminal Panel.

On the Terminal Panel, terminal type configures a virtual terminal to support the 3270 basic or extended data stream. Terminal models MOD2, MOD3, MOD4or MOD5 imply standard screen size combinations and are used with terminal type to select a logon mode table entry to be used to start sessions. An actual mode table entry (logmode) may be specified. If terminal model is not specified, screen sizes and terminal type are used to select a mode table entry.

When the SESSION command is used to start a session, screen size and terminal type keywords used on the Terminal Panel may be used in the SESSION command. The MODEL keyword and a terminal model may be used in a SESSION command to override a terminal model specified on the Terminal Panel.

TRACE

The TRACE command, abbreviated T, displays the CA Enterprise Tester Trace Panel. The Trace Panel is used to start a trace of session activity on a port. While a session trace is active, the TRACE command may be used to specify a caption for subsequently traced screen images.

Example:

```
COMMAND ===> t 2
```

Format:

```
TRACE ( options )
```

Options may specify a port number when entered at the Control Panel to start a trace. A caption may be specified when the TRACE command is entered at a Session Panel or Monitor Panel on a port on which a trace is active.

The TRACE command can be entered at the Control Panel to display the Trace Panel, in order to initiate a trace of subsequent session activity on a specified port.

The TRACE command can be entered via the Session Panel or Monitor Panel to start a trace of session activity on the current port. The TRACE command then displays the Trace Panel.

While a trace is active on a port, the TRACE command may be entered at the Session or Monitor panels to specify a caption for subsequently traced screen images. For example:

```
COMMAND ===> t 'this is a screen caption'
```

A session trace captures screen images in one of several optional formats, whenever messages are sent or received on a traced CA Enterprise Tester session. Session traces may be created for documentation or may be analyzed using a CA Enterprise Tester session analyzer.

Refer to Tracing Session Screens and Application Tests and Test Data for complete documentation of the CA Enterprise Tester trace feature and the use of CA Enterprise Tester traces to automate online system regression testing.

UP

The UP command displays previous data in the scrollable areas of the CA Enterprise Tester Control Panel and Monitor Panel, and panels which display virtual terminal screen images.

Example:

```
COMMAND ===> up max
```

Format:

```
UP ( number | PAGE |DATA   | HALF |CSR | MAX )
```

- Number specifies the number of lines to scroll;
- PAGE will scroll one page of data;
- DATA will scroll one page of data, minus one line;
- HALF will scroll one-half page of data;
- CSR will scroll to the position of the cursor within the data;
- MAX will scroll to the top margin of the data.

If an amount is not specified, the scroll amount is taken from the panel's SCROLL amount field.

The Upend DOWN commands may be used to scroll through the list of ports displayed on the CA Enterprise Tester Control Panel.

The UP, DOWN, LEFT and RIGHT commands may be used to scroll within a virtual terminal screen image when the image is larger than the terminal screen being used to access CA Enterprise Tester.

The UP, DOWN, LEFT and RIGHT commands may be used to scroll through the CA Enterprise Tester Monitor Panel's Activity Log.

UTILITY

The UTILITY command, abbreviated UT, displays the Utility Menu. The Utility Menu displays current CA Enterprise Tester utility options and allows selection of utility functions.

Example:

```
COMMAND ===> ut
```

Format:

```
UTILITY
```

For a complete description of CA Enterprise Tester utilities, please refer to CA Enterprise Tester Utilities.

VIEW

The VIEW command, abbreviated V, may be used to start or resume CA Enterprise Tester Interactive System functions which display virtual terminal screen images. These functions include playing a script which drives an online system session, session analyzer scripts and a CA Enterprise Tester Browse of a trace list or trace data set.

Example:

```
COMMAND ===> v
```

Format:

```
VIEW ( options )
```

Options may specify free-form play specifications when entered at the Play Panel to start a script.

The VIEW command may be entered at the CA Enterprise Tester Analyze Panel to start a session analyzer, may be entered at the Play Panel to start the execution of a script, may be entered at the Browse Panel to initiate a session movie, or may be entered at the CA Enterprise Tester Control Panel.

When entered at the Control Panel the VIEW command may specify the name of the data set and/or PDS member to be played as free-form operands. The VIEW command may also specify script parameters by enclosing the data in quotes. Specifications not entered as free-form operands are obtained from the appropriate panel fields. Refer to Free-Form Entry of PLAY Specifications for a description of command stacking.

The VIEW command may be entered at a Session Panel or a Monitor Panel to resume a session analysis or played script which displays virtual terminal screen images.

The VIEW command requests that a session analysis or played script be started or resumed, that screen images be displayed, and that the script is to proceed until a session mismatch is detected or until the script stops or ends.

The terminal keyboard remains locked until the function stops or ends. If a non-zero time interval has been specified for the step display time option provided on the CA Enterprise Tester Options Panel, then session steps created by a session analyzer or a played script are displayed for the specified time interval, before automatically resuming the active function and proceeding to the next session step. If a zero step display time interval has been specified, then session steps are not displayed.

The VIEW command completes when a script stops, when Monitor Panel stop options are satisfied, or when the browse or other active function ends.

When a session mismatch is detected or if a script stops for any reason, then the reason for the stop is presented via the CA Enterprise Tester Monitor Panel.

The VIEW command is a variation of the WAIT command supported in previous releases.

WAIT

The WAIT command, abbreviated W, may be used to start or resume a session analyzer or played script, to wait for a session analyzer mismatch, a played script to stop or end, or to initiate a browse of a trace data set.

Example:

```
COMMAND ===> w
```

Format:

```
WAIT ( options )
```

Options may specify free-form play specifications when entered at the Play Panel to start a script.

Additional Information:

The WAIT command may be entered at the CA Enterprise Tester Analyze Panel to start a session analyzer or may be entered at the Play Panel to start the execution of a script.

When entered at the Play Panel, the WAIT command may specify the name of the data set and/or PDS member to be played as free-form operands of the WAIT command. The WAIT command may also specify script parameters by enclosing the data in quotes. Play specifications not entered as free-form operands are obtained from Play Panel fields.

The WAIT command may be entered at a Session Panel or a Monitor Panel to resume a session analysis or played script.

The WAIT command requests that a session analysis or played script be started or resumed and that the resumed function is to proceed until a session mismatch is detected or until a played script stops or ends.

The terminal keyboard remains locked until the resumed function stops or ends. If a non-zero time interval has been specified for the step display time option provided on the CA Enterprise Tester Options Panel, then session steps created by a session analyzer or a played script are displayed for the specified time interval, before automatically resuming the active function and proceeding to the next session step. If a zero step display time interval has been specified, then session steps are not displayed.

The WAIT command completes when a script stops, when Monitor Panel stop options are satisfied, or when the active function ends.

When a session mismatch is detected or if a script stops for any reason, then the reason for the stop is presented via the CA Enterprise Tester Monitor Panel.

The VIEW command is also provided as a logical variation of the WAIT command.

ZOOM

The ZOOM command, abbreviated Z, switches between ISPF and TSO managed session displays. An ISPF managed session display presents a CA Enterprise Tester session in a window of an ISPF panel, allows the session to be used in ISPF split-screen mode, and is referred to as Window Mode. A TSO managed session display presents a CA Enterprise Tester session using the full terminal screen and is referred to as a Zoom Mode session display.

Example:

```
COMMAND ===> z
```

Format:

```
ZOOM
```

The ZOOM command may be entered while viewing a CA Enterprise Tester session via CA Enterprise Tester's ISPF Session Panel in Window Mode, to re-display the session using the full terminal screen. The Zoom Mode display eliminates required ISPF title, command and scroll amount fields, allowing the full CA Enterprise Tester virtual terminal screen to be viewed at one time.

The ZOOM command may be entered via a PF key, while viewing a CA Enterprise Tester session in Zoom Mode. The command will initiate Window Mode, and re-display the virtual terminal screen through a window in CA Enterprise Tester's ISPF Session Panel. A CA Enterprise Tester session being viewed in Zoom Mode may be re-displayed via the CA Enterprise Tester Session Panel in Window Mode when the terminal's PA2 key is pressed, based on an option provided on the CA Enterprise Tester Options Panel.

The initial session display mode used when sessions are started are determined by two options provided on the CA Enterprise Tester Options Panel.

One option may be used to specify the initial session display mode for sessions started using the SESSION command.

A separate option may be used to specify the initial session display mode for sessions started by session analyzers or played scripts.

The CA Enterprise Tester Options Panel and all user-defined CA Enterprise Tester options are described in Options and Controls.

The operation of the ZOOM command is described and illustrated in Starting Virtual Terminal Sessions.

Chapter 4: Password Security

This chapter documents features provided by CA Enterprise Tester to permit the capture and use of procedures used to sign on to online systems, including data which represents secure passwords, and ensure that actual passwords remain confidential.

- Password Security
- Token Security
- Code Security

Password Security

CA Enterprise Tester provides virtual terminals that emulate 3270 terminals. To an online system or a CA Enterprise Tester Interactive System user, a CA Enterprise Tester session that connects a virtual terminal with an online system is the same as any other terminal session. The security of the data and other resources managed by an online system is the responsibility of the system, regardless of what kind of terminal is used to access the resources. When terminal sessions are used to access system resources that require user authority, the terminal user must first sign on by entering a user ID that identifies a person authorized to access the resources, and by entering the correct password to prove that they are the identified person.

When a CA Enterprise Tester session is used by a person, passwords are entered in confidential non-display screen fields by the person in the same way they are entered using a real terminal session. However, CA Enterprise Tester is not just a terminal emulator and a CA Enterprise Tester session user is not always a person. CA Enterprise Tester sessions can be captured as CA Enterprise Tester scripts, REXX execs or screen traces, and captured sessions can be repeated to perform regression testing automatically. CA Enterprise Tester scripts or REXX execs can create CA Enterprise Tester sessions that connect batch jobs to online systems, and can use any available online applications, to perform or automate almost any imaginable terminal task that could by performed by a person using a terminal. Like any other terminal user, a CA Enterprise Tester script or REXX exec must sign on to access system resources that require user authority.

If passwords were captured in recorded or traced session data or if passwords were coded in user-written scripts or execs, then CA Enterprise Tester libraries would contain visible passwords that could be used to gain unauthorized access to protected system resources. Other methods are required to ensure that passwords captured and used by CA Enterprise Tester applications remain confidential.

CA Enterprise Tester provides two password security options that allow passwords to be captured and used by CA Enterprise Tester applications while ensuring that they are not disclosed to or used by anyone other than their owners.

- Token Security captures symbolic password tokens instead of passwords when CA Enterprise Tester sessions are captured, and allows CA Enterprise Tester applications to enter tokens instead of passwords to sign on. Token Security explains how tokens work and how they are used in CA Enterprise Tester applications.
- Code Security captures encoded password codes instead of passwords when CA Enterprise Tester sessions are captured, and allows CA Enterprise Tester applications to enter codes instead of passwords to sign on. Code Security explains how codes work and how they are used in CA Enterprise Tester applications.

The CA Enterprise Tester Interactive System SECURE command may be entered at the CA Enterprise Tester Control Panel to display the active password security option and any other options allowed by CA Enterprise Tester system options.

Token Security

Token Security is a password encryption system that allows CA Enterprise Tester applications to capture and use passwords while ensuring that they are not disclosed to or used by anyone other than their owner.

Token Security works with RACF, ACF2, CA Top Secret or any other security system that reliably establishes the user ID that owns address spaces in which CA Enterprise Tester is used, in the manner prescribed by the z/OS System Authorization Facility (SAF).

- A password is any value entered in a non-display screen field.
- A token is a name that may be used in CA Enterprise Tester applications to enter a user's password.

Token Security may be used to capture tokens instead of passwords when CA Enterprise Tester sessions are recorded or traced.

When a CA Enterprise Tester session is captured using Token Security and a password is entered, the password is replaced by a token in the captured session data. The token captured is either an existing token for the password found in the user's profile or a new token that is automatically created for the password.

When a new token is created for a user:

- A unique name is selected to be used as the new token.

- A profile is created for the user, if needed, to store private data.
- The password is combined with the user ID that owns the address space.
- The combined password and user ID is encrypted.
- The new token and the encrypted data are stored in the user's profile.
- A token library is created for the user, if needed, to store token screens.
- An image of the screen that contains the password field is stored as a token screen.
- If the password was entered by a person and system options specify password capture notification, the user is notified that a password was captured.

A TOKEN footnote identifies each token captured in a traced screen image. The footnote in the following example identifies a captured token, UID1A, at screen row 8, column 20.

```
+Session Step(PT01 ENTER--> TSOA003) Cursor(8,26) - -+
| ------------------------------ TSO/E LOGON -----   |
|                                                    |
| Enter LOGON parameters below:                      |
| Userid  ===> UID2                                  |
| Password  ===> UID1A                               |
+-TOKEN AT <8 20> --------------------------------  -+
```

The TYPE statement TOKEN keyword identifies a token in a captured or user-written CA Enterprise Tester script. For example:

```
TYPE <8,20> TOKEN 'UID1B'
<8,26> ENTER
```

The CA Enterprise Tester TYPE command TOKEN keyword identifies a token in a captured or user-written REXX exec. For example:

```
" TYPE <8,20>   TOKEN 'UID1C' ",
"  <8,24>   ENTER "
```

When a CA Enterprise Tester script, REXX exec or session trace is used to create a session and a token is used to enter a password, the user ID that owns the address space is used to decrypt the token data retrieved from the user's profile. The correct password can be recovered from the token data only in an address space owned by the user id that was used to encrypt the password.

Token Security not only allows CA Enterprise Tester applications to enter correct passwords that are not seen in scripts, execs or traces, but also ensures that passwords represented by tokens are used only by their owners.

Token Security provides other safeguards that protect passwords from accidental disclosure, protect passwords sent by online systems to terminals, and protect passwords while they are in memory.

- To prevent accidental disclosure of passwords that could occur if tokens were incorrectly typed in display fields, the PSL TYPE statement and the CA Enterprise Tester TYPE command do not allow tokens to be typed into display fields.
- Some widely-used security systems and session managers rely on 3270 non-display fields to hide displayed passwords and use them not only as password input fields, but also use both modifiable and protected non-display fields for other purposes. For example:
  - A session manager that supports concurrent virtual sessions with multiple online systems remembers passwords when users sign on to the session manager. Subsequently, when a user selects a system from a session manager menu, the session manager starts a virtual session with the system. When the selected system sends a sign on display to the virtual terminal, the session manager fills in the non-display password input field for the user before sending the sign on display to the real terminal, allowing the user to simply press the enter key to sign on to selected systems.
  - One security system is known to save passwords in protected non-display fields. When a user enters a current password and a new password to change passwords, the system sends a verify-password display that saves the user's password in a protected non-display field. When the new password is re-entered by the user, both the saved password and the re-entered password are sent to the system. Like many security systems, the system views all passwords as uppercase characters, and the password saved in the protected non-display field reflects its uppercase standard, even though it was entered by the user as lowercase characters.
  - When a user enters a password to sign on, another security system responds by sending only a protected non-display attribute that replaces the modifiable non-display attribute of the password input field and leaves the password in the display.

  When CA Enterprise Tester sessions are traced, screen images are captured that show each request sent by a user to a system and each response sent by the system to the terminal. To protect all passwords in captured data regardless of who sent them, Token Security replaces all data in modifiable non-display fields with tokens, and replaces data in protected non-display fields with tokens, if the data or a lower case version of the data matches a password referred to by any existing user token.

Protected non-display fields are commonly used by online applications to save data that is not confidential. Token Security for protected non-display fields protects passwords, but allows other application data saved in protected non-display fields to be captured and used to thoroughly analyze captured sessions when they are repeated to perform regression testing.

- Token Security protects passwords not only in scripts, execs and traces, but also in memory. Password fields are non-display even to the scripts or execs that have access to the virtual terminal display buffers that contain the fields. For example, if a token is typed into a non-display field, but is not entered, the password should be present in the field. However, if a script uses the SET statement to copy the field value to a variable, or a REXX exec assigns a symbol to the substring of PTEBUFF that represents the password field, the resulting PSL variable or REXX symbol contains only nulls. If the script or exec then types ENTER, the previously typed password is sent.

Token Security does not prevent the use of password codes and can not prevent people from coding real passwords. However, if codes or passwords are used while Token Security is active, their values are secured in memory when CA Enterprise Tester first sees them. Token Security captures only tokens regardless of how passwords are entered by CA Enterprise Tester applications.

CA Enterprise Tester Token Security Panel

The Token Security Panel is used to change the value of tokens and to create or delete tokens.

When a token is created, the token and encrypted password are stored in the user's profile. When a CA Enterprise Tester application uses a token to enter a password, the token is used to locate the password in the user's profile. Most security systems require passwords to be changed periodically. If a password is changed, and a token is used to enter the old password, then the token value (password) must be changed to the new password using the Token Security Panel.

A token screen is a screen image that contains a modifiable non-display field and is used to maintain a token and token value in a user's profile. One token screen is required to maintain each token in a user's profile. Each token screen is stored as a member of a token library and the name of each member is the token that is maintained using the token screen. When a token and token value are stored in a user's profile, an image of the screen used to enter the value is stored as a token screen in the user's token library. A token library is created for a user automatically if it does not already exist.

The Token Security Panel is used to display a specified token screen or to display a token library member list from which one or more token screens may be displayed by selecting members.

The Token Security Panel, illustrated in the following example, is displayed when the SECURE command is entered at the CA Enterprise Tester Control Panel and the active password capture format is Token, or when the TOKEN command is entered at another CA Enterprise Tester security panel.

```
+-------------------------------------------------------------------------------+
| ProTerm Token Security Panel ------------------------------------------------|
| OPTION  ===>                                                                  |
|                                                                               |
|     ENTER COMMAND/DATA   C CODE PANEL                                         |
| PF15 END PANEL DISPLAY                                                        |
|                                                                               |
|To change token value or to create or delete a token, specify token screen:    |
|                                                                               |
| Token Library    ===> 'UID1.PROTERM.TOKEN'                                    |
| Member Name      ===>         ( Blank / pattern for member selection list )   |
|      ( Select codes: S Select, B Browse, E Edit )                             |
|                                                                               |
| Active password security:                                                     |
|                                                                               |
| Capture Format  ===> Token  ( Options: Token, Code )                          |
| Userid  . . . . . . . UID1                                                    |
+-------------------------------------------------------------------------------+
```

Token Library

The user's token library is displayed when the panel is entered. Another library may be entered to display other token screens.

Member Name

A member name may be entered to display a specific token screen, or leave the field blank or enter a member name pattern to display a Token Library member list.

Capture Format

The active password capture format is displayed. If options are listed following the Capture Format field, a listed option may be entered to change the active password capture format. The active password capture format is stored in your profile and controls how passwords are captured by any CA Enterprise Tester application running under your user ID, even batch jobs.

Token Security Panel Commands

CA Enterprise Tester processing available at the Token Panel includes the following:

- The END command ignores other input, exits the Token Security Panel and returns to the Control Panel.
- If other security panel commands are listed and a command is entered, other input is ignored and the specified security panel is displayed.
- If the active password capture format is changed, other input is ignored and the security panel for the new capture format is displayed.

- The Enter key displays the token screen in the token library and member entered, or displays a token library member selection list if a member name is not entered or a member name pattern is entered.

  If a member selection list is displayed, any number of member list select codes, S to display a token screen, B to browse a member, or E to edit a member, may be entered to serially display, browse or edit each member.

Maintaining Tokens and Passwords

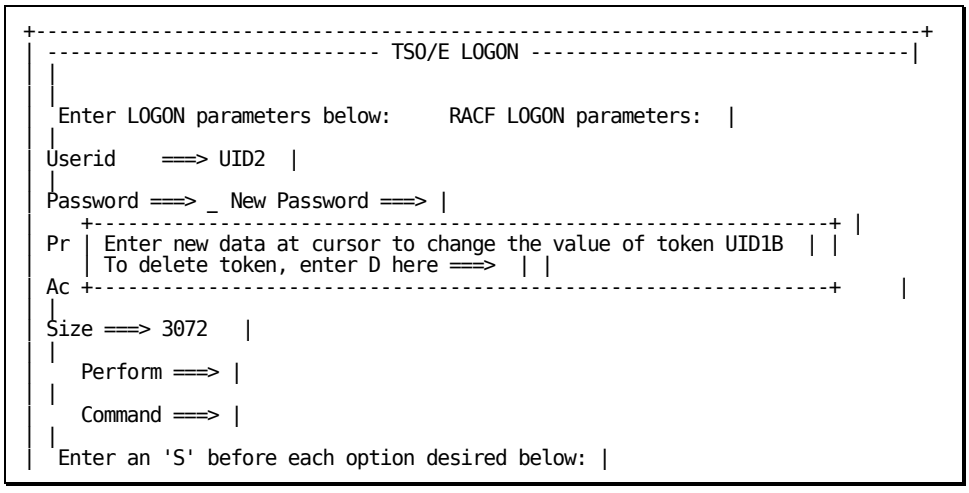
When a token library and member name are entered at the Token Security Panel, or a member name is selected from a token library member list, the token screen and an options window that partially overlays the token screen is displayed and the cursor is positioned at a modifiable non-display field. The following example shows a token screen and options window that explains maintenance options for token UID1B.

The options window explains token maintenance options depending on the status of the token and token screen.

- If the token is in your profile, a new password may be entered to change the password referred to by the token.
- If the token is not in your profile, a password may be entered to create a token. When a token is created, the token and encrypted password are stored in your profile.
- If the token is in your profile or the token screen is in your token library, they may be deleted, but only from your profile and/or token library.

If a password or delete request is entered, a confirm request window is displayed in the place of the options window and explains how to confirm your request.

The END key may be used to exit any token screen display.



```
| S -Nomail   S -Nonotice  -Reconnect    -OIDcard  |
| |
| PF1/PF13 ==> Help   PF3/PF15 ==> Logoff  PA1 ==> Attention    PA2 ==> Reshow|
|You may request specific help information by entering '?' in any entry field|
+-------------------------------------------------------------------------------+
```

Sharing User IDs and Tokens

If a user ID and password are entered to sign on while capturing a CA Enterprise Tester session, the user ID and a token for the password are captured in the recorded script or exec or in the traced screen images, and the token and encrypted password are stored in the user's profile. A script, exec or trace that contains an actual user ID and a token works only when the token is found in the user's profile. Normally, the password for a user ID is known only by one person and CA Enterprise Tester applications that contain a token for the password can be used only by that person.

However, anyone who knows the password for a user ID can use an application that contains the user ID and a token for the password by storing the token and encrypted password in their profile. While it unusual for more than one person to know the password for a user ID, system test groups or quality assurance departments sometimes use shared user IDs, authorized to perform functions in test systems only, and all members of the group know the passwords.

If shared user IDs are used by a group of people to test systems or for some other purpose, sharable scripts, execs and traces that contain user IDs and tokens can be created by entering shared user IDs and passwords when CA Enterprise Tester sessions are recorded or traced. When a shared user ID and password are entered while capturing a session, a token and encrypted password are stored in the user's profile, and a token screen is stored in the user's token library.

Any group member who knows the passwords for the shared user IDs may use any captured script, exec or trace captured by any other member of the group by using the Token Security Panel to display another group members token screens and by entering the correct passwords for the shared user IDs to store the tokens and encrypted passwords in their profile.

Token Security Subroutine

A token security subroutine is provided that may be called by any program that supports standard z/OS CALL conventions to assign a new value to a named token that belongs to the user id that owns the address space in which the token maintenance routine is called. The token security subroutine may be called by user written programs to maintain password token values automatically instead of using the CA Enterprise Tester Token Security Panel to maintain token values manually when user passwords are changed.

The token maintenance routine may be called by a program written in any language that supports standard z/OS call conventions. A call to the token maintenance routine must specify one parameter that contains the following data, expressed in COBOL:

```
77  PTEIS   PIC X(8) VALUE 'PTEIS'.
01  CALL-PARM.
   03 CALLFUNC   PIC X(8) VALUE 'ENCRYPT'.
   03 TOKENAME   PIC X(8).
   03 FIELDLEN   PIC S9(4) COMP VALUE 8.
   03 FIELDATA   PIC X(8).
```

Token Security Call Parameters

- The value of CALLFUNC is an 8 character value, 'ENCRYPT ', and specifies that the data in FIELDATA whose length is the value of FIELDLEN is to be encrypted for use by the user ID that owns the address space in which an ENCRYPT call is issued, and is to be stored as the value of the CA Enterprise Tester Security Token name specified as TOKENAME.
- The value of TOKENAME may be any 1 through 8 character left justified token name that is a valid PDS member name.
- The value of FIELDLEN is the length of the non-display password field into which the token data specified in FIELDATA is to subsequently be typed. If a password to be encrypted is shorter than FIELDLEN, then the value of FIELDATA is the password padded with blanks or nulls, as required, up to the length specified in FIELDLEN.
- The value of FIELDATA is the exact field value to be represented by the token and may be any valid field length specified by FIELDLEN.

For example, the following statements may be used to assign or change the value of a token named 'UID1PWD' to a new password value 'newpwd', that is entered in an 8 character non-display screen field:

```
MOVE 'UID1PWD' TO TOKENAME.
MOVE 'newpwd'  TO FIELDATA.
CALL PTEIS USING CALL-PARM.
```

If the return code is 0 (zero), then the encrypted token value was successfully stored as the value of the specified token.

Code Security

Code Security is a password encoding system that allows CA Enterprise Tester applications to capture and use passwords while ensuring that they are not disclosed to or used by anyone other than their owner.

Code Security provides effective password protection when:

- A security system is active that reliably establishes the user ID that owns address spaces in which CA Enterprise Tester is used, in the manner prescribed by the z/OS System Authorization Facility (SAF).
- User IDs are 2 through 8 contiguous national characters (EBCDIC A-Z, 0-9, @, #, and $).
- Passwords are 3 through 8 contiguous national characters, may be entered as lowercase characters, and are the first non-null data entered into non-display screen fields.

When a CA Enterprise Tester session is captured using Code Security and a password is entered, the password is replaced by a code in the captured session data. A code is the encoded value of a password combined with the user ID of the person who owns the password.

A footnote identifies traced screen images that contains codes. In addition to the word, ENCODED, and if allowed by system options, the footnote also documents the code owner's user ID. In the following example, 'F2P@Z', is a password code.

```
+Session Step(PT01 ENTER--> TSOA003) Cursor(8,25) ----+
| ------------------------------ TSO/E LOGON -----    |
|                                                     |
|   Enter LOGON parameters below:                     |
|   Userid    ==> UID2                                |
|   Password  ===> F2P@Z                              |
|                                                     |
+- ENCODED FOR UID1 ----------------------------------+
```

The TYPE statement CODE keyword identifies a password code in a captured or user-written CA Enterprise Tester script. If allowed by system options, a comment documents the code owner's user ID. For example:

```
TYPE <8,20> CODE  'KTL9$H'   /* CODED FOR USERID UID1 */
<8,26> ENTER
```

The CA Enterprise Tester TYPE command CODE keyword identifies a password code in a captured or user-written REXX exec. For example:

```
" TYPE <8,20>   CODE  '1SJP'  ","
<8,24>   ENTER "
```

To prevent accidental disclosure of passwords that could occur if codes were incorrectly typed in display fields, the PSL TYPE statement and the CA Enterprise Tester TYPE command for REXX do not allow codes to be typed into display fields.

When a CA Enterprise Tester application creates a session and uses a password code to sign on, the code is decoded using the user ID that owns the address space. The correct password can be recovered from a code only in an address space owned by the owner of the code. Because password codes may be seen without disclosing the password encoded and because they can be used only by their owner, codes are not confidential information.

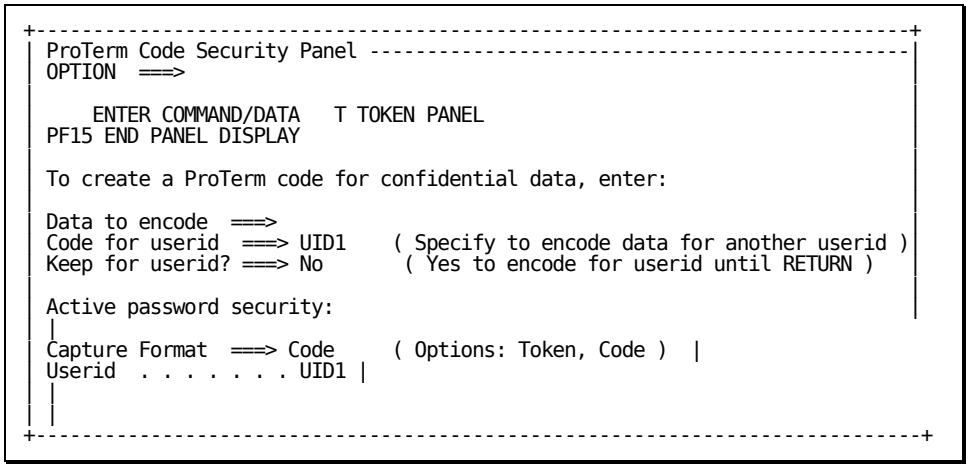
If a password, that was entered by a person, is captured as a code, and system options require password capture notification, the user is notified that a password was captured.

CA Enterprise Tester Code Security Panel

The Code Security Panel is used to create codes to be used in CA Enterprise Tester applications to enter passwords.

Because a code contains a password, if a code for a password is used by CA Enterprise Tester applications and the password is changed, the code must be replaced by a code for the new password, wherever the old code is used.

The Code Security Panel, illustrated in the following example, is displayed when the SECURE command is entered at the CA Enterprise Tester Control Panel and the active password capture format is Code, or when the CODE command is entered at another CA Enterprise Tester security panel.



Data to encode

A password or other confidential data may be entered in the non-display Data to encode field to create a code for the entered data.

Code for user ID

The Code for user ID field is available only if system options allow codes to be created for use by other user IDs. If the option is displayed, a user ID may be entered to create a code for the entered Data to encode for use by the user ID entered, or to capture a subsequent session that is encoded for use by the user ID entered.

Keep for user ID?

The Keep for user ID? field is available only if system options allow codes to be created for use by other user IDs. If the option is displayed, Yes may be entered to continue (keep) using the entered Code for user ID, until the end of the current CA Enterprise Tester Interactive System dialog. The option may be used to trace or record a subsequent session that contains codes that are encoded for the entered Code for user ID.

Capture Format

The active password capture format is displayed. If options are listed following the Capture Format field, a listed option may be entered to change the active password capture format. The active password capture format is stored in your profile and controls how passwords are captured by any CA Enterprise Tester application running under your user ID, even batch jobs.

Code Security Panel Commands

CA Enterprise Tester processing available at the Code Security Panel includes the following:

- The END command may be entered to exit the Code Security Panel and to return to the Control Panel.
- If other security panel commands are listed and a command is entered, other input is ignored and the specified security panel is displayed.
- If the active password capture format is changed, other input is ignored and the security panel for the new capture format is displayed.
- The Enter key creates and displays a password code that is the encoded value of the data entered in the Data to encode field combined with either your user ID or a user ID entered in the Code for user ID field, if that is an option. The code works correctly only when used in address spaces owned by the encoded user ID.

Code Security Subroutines

CA Enterprise Tester provides two interfaces that allow user-written programs or REXX execs to create password codes. The interfaces may be used to automate or simplify password code maintenance. CA Enterprise Tester system options may not allow codes to be created for use by other user IDs or may limit the number of codes that can be created by a single user in a single day.

The FUNCTION Interface

The PTENCODE function may be used by a REXX exec to create password codes. When a valid user ID and password are specified as arguments, the following exec creates and displays a password code.

```
/* This REXX exec says 'the code is xxxx.'    */
arg userid password
say "The code is'"PTENCODE(userid,password)"'."
```

The CALL Interface

The PTEIS ENCODE subroutine may be called by a program written in any language that supports standard z/OS CALL conventions, to create codes for given passwords and user IDs. The required parameter, response area, and procedure for COBOL are shown in the following example:

```
77  PTEIS  PIC X(9)  VALUE 'PTEIS   '. .
01  ENCODE-PARM.
   03 ENCODE PIC X(8)  VALUE 'ENCODE  '.
   03 USERID PIC X(8).
   03 CODE-LENGTH  PIC S9(4) COMP.
   03 PASSWORD  PIC X(8).
   03 CODE  REDEFINES PASSWORD PIC X(8). .
   MOVE 'USERID1' TO USERID.
   MOVE 'PASSWD1' TO PASSWORD.
   CALL PTEIS USING ENCODE-PARM.
```

If USERID and PASSWORD are valid, the password code is returned in CODE, the length of the code is returned in CODE-LENGTH, and the return code is set to 0 (zero), otherwise CODE-LENGTH is set to 0(zero), and the return code is set to 8.

Chapter 5: Script Language

This chapter may be used as a primer to learn how to write CA Enterprise Tester scripts. The CA Enterprise Tester Script Language is an easy-to-use high-level language that provides unique statements that allow CA Enterprise Tester scripts to start and drive VTAM sessions with online systems. CA Enterprise Tester scripts may be played using the CA Enterprise Tester Interactive System or may be executed using z/OS batch jobs. CA Enterprise Tester scripts may be used to automate any procedure that can be accomplished by a person using a terminal session with an online system

- The CA Enterprise Tester Script Language
- Batch JCL
- Script Language Overview
- CA Enterprise Tester Data Items
- Session Control
- Terminal Operation
- Structuring Script Code
- Variable Memory
- Data Set Access
- Message Logging
- Operator Communication
- Session Traces and Recordings
- Condition Handling and Diagnostics Aids
- Session Flows

The CA Enterprise Tester Script Language

CA Enterprise Tester provides a high-level general purpose Script Language. The CA Enterprise Tester Script Language (PSL), is a procedural language that supports verbs, working storage, structured programming techniques and other concepts that are familiar to users of other high-level languages. CA Enterprise Tester scripts are not compiled. They may be executed using the Interactive System PLAY Command or may be executed in batch jobs using the CA Enterprise Tester Script Language processor.

Using unique PSL verbs that support the emulation and operation of 3270-compatible terminals, CA Enterprise Tester scripts are able to establish sessions and communicate with any VTAM online system. CA Enterprise Tester scripts may be used to automate virtually any manual terminal operator procedure or task. PSL verbs that support communication describe the things that a person would do to start a terminal session, to enter transactions, and to examine online system responses.

A simple example will illustrate PSL communication capabilities and how CA Enterprise Tester scripts may be used to automate terminal tasks.

A classic problem at many computer installations is coordinating batch and online access to an online data set when nightly offline file maintenance is performed. A common and simple solution to this problem is for an operator to logon to the online system and close the data set using an online transaction, prior to executing the nightly batch job. After the batch job has completed, the operator would again logon to the online system and re-open the data set. This may be acceptable for a single online system or for a few data sets. For multiple systems and more than a few data sets the procedure becomes a time-consuming and error-prone task.

The automation of a manual operator procedure may be accomplished by coding a CA Enterprise Tester script that describes the steps performed in the manual procedure. Scripts may be executed as steps in a batch file maintenance job. Table 2 compares a manual operator procedure and a CA Enterprise Tester script. The CEMT transaction used in these scripts is a standard system control transaction distributed as a part of the CICS product. Both the manual procedure and the CA Enterprise Tester script solve the problem of opening a data set for use by online system applications at the completion of an offline file maintenance job. The CA Enterprise Tester script accomplishes the task automatically.

The script shown in Figure 5-1 works, but much more could be done by the script. Closing CICS Data Sets from a Batch Job Step on page A-2 and Opening CICS Data Sets from a Batch Job Step on page A-3 illustrate general purpose scripts that use other PSL features and that provide improved user options and flexibility.

The simplicity and power of the CA Enterprise Tester Script Language makes it easy to automate any manual procedure that requires an operator to enter online system transactions using a 3270-compatible terminal.

Manual Operator Procedures	Automated CA Enterprise Tester Script
Wait for the nightly batch job to end	* automatically runs as last step * of nightly batch job
Log onto the CICS system via a 3270 CRT	LOGON CICS1
Type: CEMT S DA(USERFILE) OPE and press the ENTER key.	OPEN TYPE 'CMET S DA(USERFILE) OPE'ENTER
Make sure that the CEMT response at screen row 3,column 64, is 'NORMAL'	If <3,64,*> NE 'NORMAL' THEN GO TO ERROR ELSE TYPE PF3
Press PF# to terminate the CEMT transaction	ENDIF END
End the session with CICS	ERROR WTO "'USERFILE'OPEN FAILED"
In the event of open failure determine the reason for failure and correct, or call programmer for help	WTOR "REPLY 'RETRY'OR 'ABEND'" IF &PTEREPLY EQ 'RETRY' THEN GO TO OPEN ELSE ABENDENDIF

The automation of online system control is only one application of CA Enterprise Tester scripts. CA Enterprise Tester scripts have virtually unlimited applications.

- CA Enterprise Tester scripts may be used to read data sets that are uploaded from PC's and other distributed processors and to enter the data into mainframe online systems using existing data entry transactions.
- Scripts may be used to perform mass updates or corrections to an entire database, without the need to take the database offline.
- By exploiting the logic contained in existing online transactions, complex routines need not be reproduced for one-time procedures and uniform validation and processing of all maintenance is assured.
- Because scripts exploit existing online software, audit trails built into production online systems are implemented automatically.
- Scripts may be used to perform online system regression testing.
- Multiple scripts may be executed concurrently to perform online system integration and stress testing.

The CA Enterprise Tester Script Language provides many powerful functions and features. This chapter may be used as a primer to learn how to write and use CA Enterprise Tester scripts. The chapter begins with sections that illustrate the basic JCL required to execute CA Enterprise Tester scripts, and that describe general language rules and the syntax of CA Enterprise Tester statements and data items. An overview of major language capabilities is also provided.

Subsequent sections of this chapter describe groups of related language statements that support major language capabilities, including session management, virtual terminal operation, structured programming techniques, variable memory, data set access, error handling other features.

PSL Reference provides detailed documentation of the function, syntax and options supported by each PSL verb. The sections are arranged in alphabetic order by verb name for easy access.

Batch JCL

The CA Enterprise Tester Script Language processor, program PTEBS, may be executed as a batch job step to process a CA Enterprise Tester script, using the following JCL:

```
//jobname JOB . . .
//stepname EXEC PGM=PTEBS,REGION=1024K,PARM='parameters'
//PTEIN   DD DSN=PTE.SCRIPT(script),DISP=SHR
//PTELIB DD DSN=PTE.SCRIPT,DISP=SHR  optional
//PTELOG DD SYSOUT= or DSN=
//PTETRACE DD SYSOUT= or DSN=   optional
//PTEREC DD SYSOUT= or DSN=   optional
```

REGION Size

REGION in bytes may be approximated as follows:

```
REGION = 256000 + (80 x (number of script statements))
 + any large tables created by scripts
 + data set buffers
```

PARM - Passed Parameter Data Area

PARM may specify one or more keywords that specify script processor options. Supported keyword options are as follows:

Keyword	Description
COMMA	COMMA specifies that a comma is to be used as a decimal point. When a comma is used as a decimal point, periods may be used to punctuate the integer part of a number. The default value for the decimal point is based on a system installation parameter and may be either a comma or a period.
FLOW	FLOW specifies that each script statement is to be logged as it is about to be executed. NOFLOW, the default, specifies that script statements are not to be logged as they are executed.
HRESP	HRESP may be used to turn response handling on as though a HANDLE RESPONSES ON statement had been executed prior to executing the script. For a complete description of the Script Language HANDLE RESPONSES statement, refer to RESPONSES on page 6-49.
LIST	LIST specifies that a listing of executed scripts should be written to the PTELOGDD statement as they are being loaded for execution. NOLIST, the default, specifies that a script listing is not to be produced.
PERIOD	PERIOD specifies that a pexriod is to be used as a decimal point. When a period is used as a decimal point, commas may be used to punctuate the integer part of a number. The default value for the decimal point is based on a system installation parameter and may be either a comma or a period.
PRECORD	PRECORD specifies that the ISPF packed data format is to be used when a script is written to the PTEREC DD statement. NOPRECORD specifies that scripts are not to be packed. The default is controlled by a system installation option.
PSL/REXX	PSL specifies that when interactive system sessions are recorded, CA Enterprise Tester scripts in the CA Enterprise Tester Script Language should be created. REXX specifies that REXX execs, which may be executed using CA Enterprise Tester Host Command Environment, should be created. The default is controlled by a system installation option.
PTRACE	PTRACE specifies that the ISPF packed data format is to be used when screens are written to the PTETRACE DD statement. NOPTRACE specifies that screens are not to be packed. The default is controlled by a system installation option.

Keyword	Description
RCHG/RMDT	RCHG specifies when a recording is active, only screen fields which are changed, excluding unchanged trailing null data, should be recorded. RMDT specifies that all MDT fields,(modified data tag), including MDT field data are to be recorded. The default is controlled by a system installation option.
RDELAY	RDELAY specifies that when a recording is active, Script Language DELAY statements are to be recorded. DELAY time intervals represent the amount of time between the last online system response and an AID key, (attention identifier), which sends a request to the system. RDELAYOFF specifies that Script Language DELAY statements are not recorded. The default is controlled by a system installation option.
RULOG	RULOG may be specified to request that SNA Request Units (RU) and messages that document session events, are to be logged. NORULOG, the default, specifies that a request unit log is not to be produced.
S522OFF	Scripts that invoke very long running transactions may incur an S522 abend, sometimes referred to as a timeout, if the script waits for a transaction response longer than an installation controlled wait time limit. S522OFF may be specified to disable S522 abends while a script is waiting on transactions known to be long running.
TCHAINS	TCHAINS, the default, specifies that when a trace is active, all outbound session screens are to be captured. (See TFREEKB below)
TDELAY	TDELAY specifies that when a trace is active, Script Language DELAY statements are to be written to the trace data set. DELAY time intervals represent the amount of time between the last online system response and an AID key, (attention identifier), which sends a request to the system. TDELAYOFF specifies that Script Language DELAY statements are not captured. The default is controlled by a system installation option.
TFREEKB	TFREEKB specifies that when a trace is active, only outbound session screens which free the virtual terminal keyboard are to be captured. (See TCHAINS above)

Keyword	Description
TLOGON	TLOGON specifies that when a trace is active, Script Language LOGON statements are to be written to the trace data set. TLOGONOFF specifies that Script Language LOGON statements are not captured. The default is controlled by a system installation option.
xTRACE	One of the keywords, FTRACE, ITRACE or UTRACE may be used to specify the type of trace to be created when screens are written to the PTETRACE DD statement. FTRACE requests a format trace, that includes attributes and unedited screen data. ITRACE, the default, requests an image trace that is edited to convert attributes and binary zeros to blanks, and graphic escape characters to dashes, (-). UTRACE requests an image trace that contains only uppercase alphabetic data.

The JCL PARM area may also be used to assign initial values to user-defined Script Language variable fields. For example, a CA Enterprise Tester script may be written to LOGON to an online system whose name is contained in a variable field, &SYSTEM. The following JCL PARM provides the name of the online system to the script:

```
//STEP1 EXEC PGM=PTEBS,PARM='SYSTEM=CICS1'
```

In the preceding example, the expression, SYSTEM=CICS1, is the equivalent of executing the following Script Language SET statement, prior to executing the first statement of the script presented to the CA Enterprise Tester Batch System via the PTEINDD statement:

```
set &system to 'CICS1'
```

Variables that are set via the JCL PARM are character variables. Variable subscripts are not recognized in PARM data. All data following 'VARIABLE=', to the next comma or the end of PARM data is assigned to the named variable.

Any number of keyword options and/or variable field values that can be contained in a 100-character string may be specified. Parameters are separated and delimited by a comma. Spaces may be imbedded within the variable field value.

For example:

```
// EXEC PGM=PTEBS,PARM='NOLIST,SYS=CICS1,OPT=OPEN,DD=ADDN'
```

A parameter list may span multiple records using standard JCL continuation syntax. For example:

```
// EXEC PGM=PTEBS,PARM=('NOLIST,NOFLOW,S522OFF',
// 'SYS=CICST1,OPT=ORGDB',
// 'DD1=VNDRMSTR,DD2=APTRXNS')
```

Data Set Definition Statements

Data definition, (DD), statements which begin with PTE are reserved by CA Enterprise Tester. These statements refer to required and optional data sets which may be used by a batch job executing the CA Enterprise Tester Script Language Processor, PTEBS, as well as other CA Enterprise Tester batch processing.

DDNAME	Description
PTEIN	The main script to be executed by the Script Language processor is identified by a PTEIN DD statement. The PTEIN data set must contain only fixed length 80 character records. Positions 73 through 80 of each record are reserved for a conventional sequence number field and are ignored by CA Enterprise Tester. Concatenated PTEIN data may be a mixture of packed and unpacked scripts.
PTELIB	PTELIB is an optional DD statement that is used to allocate one or more concatenated script libraries containing scripts that are executed by other scripts using the PSL LINK statement. If a PTELIBDD statement is not specified or if a linked-to script is not found in libraries allocated using the PTELIB DD statement, then any script libraries referred to by the PTEIN DD statement and/or the CA Enterprise Tester system script library specified in system options are searched automatically.
PTELOG	During script execution, a running log is kept that documents significant session events and errors that are encountered. A script may also write messages to PTELOG using the Script Language LOG statement.
PTEREC	This DD statement is optional. If present, a CA Enterprise Tester script is generated that contains LOGON, TYPE, INVITE, LOGOFF, and optionally, DELAY statements that represent session activity generated by the executed script. The resulting script is identical to a script that is created using the Interactive System Record Facility. The data set specified must be a sequential data set, a PDS member or SYSOUT that has DCB RECFM=F or FB and LRECL=80. If a PTEREC DD statement is supplied, the data set is opened at the beginning of script execution and is automatically closed when script execution ends. The Script Language RECORD statement, the RECORD TO ddname statement, and the ENDRECORD statement may be used to turn a recording on and off, may be used to add script generated records to the recording, and may be used to direct recorded statements to a DD specification other than PTEREC.

DDNAME	Description
PTETRACE	This DD statement is optional. If present, each screen sent to or received from an online system is written to the data set specified. The data set specified should be a sequential data set, PDS member or SYSOUT with DCB RECFM=FB or FBA and LRECL 133 through 256. If DCB information is unspecified or unacceptable, then the preferred RECFM=FBA and LRECL=135 will be used. If possible, traced screens are placed in a box that identifies the screen's borders. If LRECL=133is specified and 132 character wide screens are traced, then the screen border will not include the box sides whose presence would cause records to exceed LRECL=133. If a PTETRACEDD statement is specified, the data set is opened at the beginning of script execution and is automatically closed when script execution ends. The Script Language TRACE statement, the TRACE TO ddname statement, and the ENDTRACE statement may be used to control the format of a traced screen image, to selectively trace screen images, to specify a caption to be included on subsequently traced screens, and may be used to direct traced screen images to a DD specification other than PTETRACE.

Script Language Overview

CA Enterprise Tester scripts are coded on 80 character fixed length records. The Script Language processor looks at data contained in positions 1 through 72 of each script record. Positions 73 through 80 of each record are ignored and may be used as a conventional record sequence number field.

Scripts are most conveniently stored as members of a standard partitioned data set, but may also be stored in a sequential data set or may be simple, DD * data.

When the Script Language processor is executed as a batch job, the script to be processed is read from the data set specified on the PTEINDD statement. Scripts may also be executed using the Interactive System by entering the PLAY Command to display the Play Panel. The Play Panel contains fields that are used to enter the data set and member containing the script to be executed. Interactive script execution supports interactive debugging and provides script monitoring facilities.

A script may execute another script using the Script Language LINK statement. Linked-to scripts are read from members of a CA Enterprise Tester Script Library. A batch job identifies Script Libraries via a PTELIB DD statement. Script Libraries are identified to scripts executed interactively via fields on the Play Panel or the Analyze Panel.

Script Language Syntax

A CA Enterprise Tester script consists of labels, statements and comments.

Script Labels

A script may contain labels that may be referred to by GOTO and PERFORM statements. Script labels are coded in position one of a script record.

Script labels may contain dual-case alphabetic, numeric and national characters, @ , # , $, and hyphens, -. A label must include at least one alphabetic or national character. A label may be any desired length, up to 71 characters.

For example:

```
0100-This-is-a-valid-$cript-statement-label
```

Script Statements

A script statement begins with a verb. A verb defines a function to be performed. Each verb defines required and optional keywords and other parameters. A statement is a verb plus the keywords and other parameters that provide the information needed to perform the function defined by the verb.

When a statement is coded, the only rule is that the elements of the statement must be coded only in positions 2 through 72 of script records. Data found in position 1 of a record is recognized as a script label and delimits a statement. A statement may span as many script records as are required to complete the statement. No indication of continuation is required. A statement may be coded on a record that begins with a statement label. Multiple statements may be coded on the same record.

Dual case alphabetic characters may be used to code verbs, keywords and other elements of a statement. Upper and lowercase alphabetic characters that are not a part of a literal value are considered to be equal.

The following labeled statement spans multiple script records. The statement keys data into a virtual terminal and presses the Enter key.

```
add-record  TYPE 'add'
tab erase '123456'
tab erase 'abcdef'
enter
```

Script Comments

CA Enterprise Tester scripts may contain comments. The Script Language supports two comment conventions.

- An asterisk, * , coded in position one of a script record, identifies the entire record as a comment. The remainder of the script record is not examined by the Script Language processor.

  For example:

  ```
  * this line is a comment
  ```

- The characters, slash asterisk, /* , may be coded anywhere within a script, except within a literal, or on a comment record, to identify the beginning of a comment. The end of the comment is identified by the characters, asterisk slash, */ , which may be coded on the same or any subsequent script record.

  For example:

  ```
  logoff-routine logoff /* this statement ends a session */
  ```

Script Language Verbs

The CA Enterprise Tester Script Language supports a variety of functions and features. Subsequent sections of this chapter describe and provide examples that show how to use Script Language statements. PSL Reference on page 6-4 contains reference material that describes the function of each Script Language verb and the keywords and parameters that are supported by each. The following summary provides an overview of the CA Enterprise Tester Script Language (PSL) and shows the groups of verbs that are designed to support major CA Enterprise Tester functions and features.

The CA Enterprise Tester Script Language defines verbs that are used to start sessions and to communicate with online systems. Verbs are provided to pace session flow and to control unusual session events.

Verb	Description
LOGON	Creates a session between a CA Enterprise Tester virtual terminal and an online system.
TYPE	Enters keystrokes an a CA Enterprise Tester virtual terminal keyboard.
INVITE	Permits an online system to initiate a transaction on a session.
DELAY	May be used to capture and reproduce actual user think time between transactions in a recorded script.
LOGOFF	Terminates a session with an online system.

Verb	Description
HANDLE	Controls how unusual session events and session errors are handled.
TIMEOUT	Specifies the maximum time to wait for an online system response.
THINK	May be used to control simulated user think time and the workload generated by one or more scripts.
WAIT	Delays script execution for a specified amount of time.

The CA Enterprise Tester Script Language provides verbs that support structured programming techniques and that may be used to execute other scripts, other script language statements, and programs written in other languages.

Verb	Description
IF-THEN-ELSE-ENDIF	Logic compares data items and structures conditionally executed script routines.
DO-WHILE-UNTIL-ENDDO	Logic structures repeated script routines and exit conditions.
PERFORM	Executes a labeled script routine that uses EXIT to return to the statement following the PERFORM.
LINK	Executes a named script in a script library that uses EXIT to return to the statement following the LINK.
GOTO	Unconditionally goes to another labeled script statement.
CALL	Executes a program written in another language.
EXECUTE	Processes a list of one or more CA Enterprise Tester data items, including literals and variable fields, which are concatenated to construct a single value interpreted as another Script Language statement for immediate execution.
END	Terminates script execution

The CA Enterprise Tester Script Language processor supports named working storage fields called variable fields. Named variables may be provided to a script using JCL or Play Panel parameters or may be created when data is read into a named variable. CA Enterprise Tester provides all kinds of information to a script via CA Enterprise Tester defined variable fields. The CA Enterprise Tester Script Language defines verbs that may be used by a script to perform a variety of operations on user defined variable data, or that simply update the value of CA Enterprise Tester defined variables.

Verb	Description
SET	Creates or alters variable fields, variable field arrays, substrings of variable fields, increments or decrements numeric variables, converts data to a desired form, or obtains a value that indicates one or more properties of other data.
DROP	Returns a variable or variable array to an initial null value state.
COMPUTE	Provides comprehensive arithmetic computational capabilities on data items which may be numbers, variables, variable occurrences, variable substrings, and screen fields using 30 digit floating point precision, including rounding and other automatic operations.
DATE and TIME	Put the current date and time in CA Enterprise Tester defined variable fields. Many other verbs set the value of CA Enterprise Tester defined variables that provide information related to the function performed.

The CA Enterprise Tester Script Language provides verbs that may be used to access sequential data sets, members of partitioned data sets, and VSAM key sequenced data sets.

Verb	Description
ALLOCATE	Allocates a data set of PDS member to a specified DD name.
OPEN	Opens a data set for read-only or write operations.
READ	Reads a record sequentially or by key, into a named variable field.
WRITE	Randomly or sequentially writes a record from variable memory.
UPDATE	Updates a record in a VSAM key sequenced data set
DELETE	Deletes one or more records from a VSAM key sequenced data set.
CLOSE	Closes a previously opened data set.
FREE	Deallocates a data set associated with a specified DD name.

CA Enterprise Tester automatically captures session screen images or records Script Language statements that capture keystrokes entered on a session, when PTETRACE, PTEREC, or other DD statements are provided in batch JCL or when interactive traces or recordings are started. The CA Enterprise Tester Script Language provides statements that may be used to control or customize traces or recordings.

Statement	Description
TRACE	May be used to specify a screen trace type, to control the packed data feature, to specify screen captions, to specify which DD to write to, or to turn trace on or off.
RECORD	May be used to write statements into a recording that supplement or document automatically recorded statements, to specify which DD to write to, or may be used to turn a recording on or off.
ENCODE	May be used to select password capture security options allowed by CA Enterprise Tester system options.

CA Enterprise Tester scripts, called session analyzers, are distributed as a part of the CA Enterprise Tester product. Session analyzers re-create previously traced CA Enterprise Tester sessions and compare live online system responses to traced responses, to detect differences. In addition to the verbs that provide the ability to create and operate terminal sessions, and the verbs that provide the ability to write plain old procedural code, the CA Enterprise Tester Script Language provides two additional verbs that give session analyzers the ability to process CA Enterprise Tester trace data sets and the ability to support simple masking when screen data is compared.

Verb	Description
TRACK	Reads a screen image from a previously created CA Enterprise Tester trace data set, and puts the screen image and other session information in screen buffers and variable fields that can be processed by a script using standard Script Language screen and variable field expressions.
MASK	Identifies or describes fields on a screen, read using the Script Language TRACK statement, that are to be considered equal when they are subsequently compared to other data, by subsequently executed Script Language statements. This capability allows simple scripts to be written that intelligently decide what to mask on a screen, without the need to re-create the logic of a session analyzer for each online application, and without the need to mark up every screen that is ever traced.

The CA Enterprise Tester Script Language provides verbs that assist in script development, problem analysis, and problem control.

Verb	Description
LOG	Writes any desired information to the CA Enterprise Tester log, (PTELOGDD).
FLOW	Controls the logging of script statements, as they are executed.
DUMP	Logs the current session screen image or the contents of the CA Enterprise Tester variable pool.
ABEND	Terminates script execution with an z/OS user ABEND.
PAUSE	Makes a script stop and wait for instructions from an Interactive System terminal user.
WTO	Writes a message to the job log, system log or system console.
WTOR	Writes a message to the system console, waits for and receives a reply.
LIMIT	Limits the number of times any individual Script Language statement may be executed.

CA Enterprise Tester Data Items

Most CA Enterprise Tester Script Language statements require or allow the use of one or more data items that are used in various ways to accomplish the function performed by a Script Language verb. The data items supported by CA Enterprise Tester include literals, actual values, numbers, variable fields including variable field occurrences and substrings, and screen fields. This section describes the standard syntax of CA Enterprise Tester data items when coded in Script Language statements.

Literal Values

Many Script Language statements require or allow the specification of literal data items. A literal data item is a data value that is enclosed in single or double quotes.

For example:

```
'This is a literal data value'
'This literal contains "double" quotes'
"This literal contains 'single' quotes"
```

A literal, including the opening and closing quote, must be contained on a single script record. In the rare event that a script must use a longer literal, a SET statement may be used to concatenate the literal value into a variable field, then the variable field may be used instead of a literal. For example:

```
set &big-literal to 'This literal value is far too big to '
  'fit on a single record, but this '
  'statement puts the value in a '
  'variable field where it may be '
  'used conveniently.'
```

When comparing literal data to some other value, the alphabetic case of the data is significant. However, options of the SET statement support case conversion so that data may be compared without regard to case.

For example, if an area displayed on a screen may be dual case character data from a file, and you wish to analyze the data in some way, then the following statements can be used to view the data displayed on the virtual terminal screen as uppercase data allowing comparison to an uppercase literal:

```
set &capital-name to capital <&row &col 8>
if &capital-name = 'MISSOURI'
then
log '"Missouri" found in capital and/or small letters'
endif
```

In some specific cases, system requirements dictate that data must be capital data only. When a data item is used for a purpose that is known to CA Enterprise Tester to have an uppercase requirement, then CA Enterprise Tester automatically capitalizes the data.

For example, z/OS data set names may include capital letters only. CA Enterprise Tester automatically capitalizes the literal value in the following statement because it knows that data set names are uppercase:

```
allocate infile dataset 'a.data.set.name' shr
```

Actual Values

In many Script Language statements, the purpose of a value is indicated by a verb or other statement keyword that precedes it. Following some keywords where CA Enterprise Tester has knowledge of the purpose of the data value, it is not necessary to enclose the data value in quotes. When documentation of a Script Language statement specifies that an actual value may be specified, it means that the data need not be enclosed in quotes. For example:

```
LOGON CICS1
OPEN SYSUT1 INPUT
```

Numbers

The CA Enterprise Tester Script Language understands and operates on decimal numbers. Decimal numbers may be used in COMPUTE, SET, IF and other Script Language statements that operate on or view data numerically. A decimal number is a real base 10 number expressed in a form recognizable by people.

A decimal number consists of:

- An optional leading sign, + or -;
- Optional spaces between a leading sign and a number;
- An optional 1 through 30 digit integer that may be punctuated to separate groups of 3 integer digits;
- An optional decimal point;
- An optional decimal fraction.

The CA Enterprise Tester Script Language supports two international conventions for the punctuation of decimal numbers. A CA Enterprise Tester install option, DECIMAL=PERIOD|COMMA, or JCL PARM keywords, PERIOD or COMMA, allow an installation or user to specify whether a decimal point is represented by a period or by a comma. The opposite character becomes a recognized integer punctuation character.

The following are examples of decimal numbers:

```
DECIMAL=PERIOD      DECIMAL=COMMA
   1                     1
  -1                    -1
  + 1                   + 1
 3.14                   3,14
 1000                  1000
1,000,000            1.000.000
.666667              ,666667
-1,234.56           - 1.234,56
```

The following examples illustrate uses of decimal numbers in a script:

```
compute &circumference = 3.14 * &diameter
if &temperature < -10
then
  log 'this is really cold'
endif
set &sales-total up by &daily-sales
if &sales-total > 1,000,000
then
log 'a million in sales has been exceeded'
endif
```

In addition to Script Language support for operations on decimal data, numbers are also used within the Script Language for special purposes. For example, numbers may be used as subscripts to refer to occurrences in a variable field array, to identify variable field substring position or length values, to specify screen row, column or screen field length values, or for other purposes defined by particular Script Language verbs.

Permitted values of decimal numbers used for special purposes are limited to values that make sense. For example, subscript, position, length, row and column values must be unsigned integers. Unsigned integers consist of only an unpunctuated integer and may not have a sign or decimal point. Particular data item syntax rules or Script Language statements may restrict the valid range of an unsigned integer that is used for a particular purpose. For example, the return code specified by a Script Language END or ABEND statement must be an unsigned integer in the range of 0 through 4095.

```
The following examples illustrate uses of unsigned integers:
set &row1 to <1 1 80>
compute &total = &entry(1) + &entry(2) + &entry(3) round 2
set &column(,1,15) from &total punctuate justify right
abend 7
```

CA Enterprise Tester Script Language statements other than the SET statement understand and operate only on decimal numbers. In order to use other kinds of numbers defined by IBM System 370 architecture, including zoned, packed and binary numbers, a script must first convert the number to decimal format using the Script Language SET statement. The SET statement may also be used to convert decimal data to zoned, packed or binary format.

- Zoned numbers consist of a string of decimal digits, (the characters 0 through 9), however, the high order 4 bits of the last digit of a zoned number are used to represent the sign of the number using 4-bit codes defined by IBM System 370 architecture. (The preferred sign codes are: X'F' or X'C' is positive; X'D' is negative).

  The Script Language SET statement TO ZONED option allows a script to convert decimal numbers to zoned format. The SET statement FROM ZONED option allows a script to convert zoned numbers to decimal format. While zoned numbers are defined as signed integer values, the SET statement PRECISION keyword allows a script to specify an implied number of decimal places to be used when zoned numbers are converted to or from decimal format.

  For example:

  ```
  set &decimal-number   from zoned precision 2 &zoned-number
  set &zoned-number(,1,10) to zoned precision 2 &decimal-number
  ```

- Packed numbers may be from 1 to 16 bytes in length and consist of a string of 4-bit decimal digits (the hexadecimal values X'0' through X'9'), and a 4-bit sign code defined by IBM System 370 architecture, as the last 4 bits of the packed number. (The preferred sign codes are: X'F' or X'C' is positive; X'D' is negative). The CA Enterprise Tester Script Language supports packed numbers that contain 1 through 30 significant 4-bit decimal digits.

  The SET statement TO PACKED option allows a script to convert decimal numbers to packed format. The SET statement FROM PACKED option allows a script to convert packed numbers to decimal format. While packed numbers are defined as signed integer values, the SET statement PRECISION keyword allows a script to specify an implied number of decimal places to be used when packed numbers are converted to or from decimal format.

  ```
  set &decimal-number   from packed precision 2 &packed-number
  set &packed-number(,1,6) to packed precision 2 &decimal-number
  ```

- Binary numbers are base 2 numbers that may be from 1 to 8 bytes in length and may be signed or unsigned. All bits of an unsigned binary number are binary digits that have a value of either 0 or 1. In a signed binary number, the left-most bit is used to represent the sign of the number. A sign of 0 is positive. A sign of 1 is negative. Positive numbers, like unsigned binary numbers, are represented in true binary notation. Negative numbers are represented in two's-complement notation as defined by IBM System 370 architecture.

  The SET statement TO BINARY option allows a script to convert decimal numbers to binary format. The SET statement FROM BINARY option allows a script to convert binary numbers to decimal format. The SET keyword, SIGNED is used to identify signed binary numbers. While binary numbers are defined as integer values, the SET statement PRECISION keyword allows a script to specify an implied number of decimal places to be used when binary numbers are converted to or from decimal format.

  For example:

  ```
  set &decimal-number from signed binary precision 2 &binary-number
  set &binary(,1,4)   to signed binary precision 2 &decimal-number
  ```

  The Script Language IF statement condition keywords, DECIMAL, ZONED and PACKED may be used by a script to determine if a record field contains data that is valid for a specified numeric format.

  For example:

  ```
  if &record(,20,10) is packed
  then   log 'record field is valid packed data'
  else   log 'record field is not valid packed data'
  endif
  ```

Variable Fields

CA Enterprise Tester provides variable memory that serves as general purpose working storage. For example, variable memory may be used to pass parameters to an executed script, may be used to remember any data, may be used as counters, accumulators, calculated results, subscripts, length values, or may be used as data set I/ O buffers. A single variable field may contain up to 32,760 bytes of data.

Variable memory is dynamic. Named variable fields are created when:

- Parameters are passed to a script via the JCL EXEC statement PARM specification;
- Parameters are passed to a script from the Interactive System Play Panel EXEC PARM data entry area;
- When a record is READ into a variable field,
- When a script uses the SET statement to assign a value to a variable.
- When a script uses the COMPUTE statement to calculate a value for a variable.

Variables may be singly occurring fields or may be arrays that have one or more dimensions. Additionally, a script may refer to a substring of a character variable. For example, a script may refer to a logical field of data within a record that was read into a variable field using the Script Language READ statement.

Numeric and Character Variables

When data items are compared using conditional expressions specified in Script Language IF and DO statements, the data items are considered to be either numbers or character strings. When the data items are numbers the comparison is based on the signed decimal values of the numbers, otherwise both data items are considered to be character strings and the comparison is based on a left-to-right comparison of corresponding bytes in the character strings.

So that variable fields can be recognized as numbers when they are compared to other data, when the value assigned to a variable is an unedited decimal value, CA Enterprise Tester remembers that the variable is a number by identifying the variable as a numeric variable otherwise the variable is identified as a character variable. When a pair of data items is compared, both data items are considered to be numbers if one of the data items is a decimal number or a numeric variable and the value of the other data item is a decimal value. If neither data item is a decimal number or a numeric variable or if the value of one of the data items is not a decimal value, then the data items are compared as character strings.

A numeric variable is a signed floating point number that may have up to 30 significant digits and its value is treated as a single value rather than a string of characters. Because the value of a numeric variable is not a string of characters, numeric variables do not have substrings.

A numeric value may be edited using a Script Language COMPUTE or SET statement that assigns a numeric value to a substring of a variable field or that specifies any of the editing keywords, SIGN, PUNCTUATE or JUSTIFY. A variable field becomes a numeric variable when it is assigned an unedited numeric value using one of the following Script Language statements:

- A COMPUTE statement is used to calculate the value of a variable field;
- A SET statement source value is a decimal number or a numeric variable that is not qualified using any SET statement adjective;
- A SET statement uses the DECIMAL adjective to describe the source or target value;
- A SET statement is used to set the value of a variable field UP or DOWN by some decimal value;
- A SET statement uses the FROM preposition and one of the adjectives, ZONED, PACKED or BINARY, to convert another kind of number to decimal format.

Variable Field Reference Syntax

In a Script Language statement, the general syntax of an expression that refers to a numeric variable is:

```
&field-name(occurrence)
```

The general syntax of an expression that refers to a character variable is:

```
&field-name(occurrence substring-position substring-length)
```

where:

- &field-name is the name of a variable field. An ampersand, & , is always the first character of a variable field name, and distinguishes variable field names from other Script Language expressions. A variable field name may include alphabetic, numeric and national characters, @, #, and $; hyphens, - ; and may be up to 71 characters long.

  The first character following the ampersand must be an alphabetic or national character. Variable fields beginning with the characters PTE are reserved for special purposes defined by CA Enterprise Tester and may not be directly modified by Script Language statements.

  The following examples illustrate some uses of variable fields:

```
read infile into &record /* read into a variable  */
set &rows to 0 /* log all rows of the   */
do while &rows le &pterows /* current ProTerm screen   */
   set &rows up 1 /* using a numeric variable */
```

```
   log <&rows 1 &ptecols> /* as a screen row index    */
enddo
```

- Optionally, information may be enclosed in parentheses, (and), immediately following a variable field name to refer to an occurrence in a variable field array or to refer to a substring of a character variable field.
- Occurrence is the first value contained within parentheses following a variable field name, and refers to a numbered entry within an array. If a variable field expression refers to a substring of a singly occurring variable field, then a zero or a comma may be coded for occurrence to indicate that the variable field expression does not refer to an occurrence in a variable field array. An occurrence in a one dimensional array may be specified by coding a subscript. A subscript may be an unsigned integer or a non-subscripted name of another variable field that contains the desired occurrence in a variable field array. An occurrence in a multiple dimensional array is denoted by coding multiple subscripts enclosed within a second set of parentheses.

  For example:

  ```
  &variable1(1) /* the 1st occurrence in an array  */
  &variablen(&n) /* the nth occurrence in an array  */
  &variable-entry((&i &j)) /* an entry in a 2 dimension array */
  ```

- Substring-position is optional and is used to refer to a specific position within a character variable field. Substring position may be expressed as an unsigned integer or may be a non-subscripted variable field that contains the desired substring position.
- Substring-length is optional and is used to specify the length of a substring of a character variable field, beginning at the specified substring position. Substring length may be expressed as an unsigned integer or may be a non-subscripted variable field that contains the length of the substring.

  For example:

  ```
  &record(,10,3)  /* a 3 character field at position 10 in a record */
  ```

  In a SET statement, when a target variable substring position is specified but a substring length is not, then the substring length is obtained from the source length. When a variable substring is used for any other purpose and substring length is not specified, then substring length defaults to the remainder of the variable field, beginning at the specified substring position.

  An asterisk, * may be specified for substring length when a variable substring is compared to another data item, to specify that the length of the other data item is to be used as the length of the substring.

Variable Arrays

A CA Enterprise Tester variable field may be referred to without the use of a subscript or may be referred to using a subscript whose value is zero. A CA Enterprise Tester variable field array is a variable field that occurs more than one time. Each entry in a variable array is referred to using a subscript that indicates the entry number in the array. Every variable field array has an entry whose subscript is zero and this entry is referred to if a subscript is not specified. A script may ignore the existence of the zero entry and refer only to occurrences 1 through n.

The number of entries in a CA Enterprise Tester variable field array is not fixed. New entries are added to an array by simply assigning a value to an entry in the array that did not previously exist using a SET or COMPUTE statement. When a new entry is added to an array by assigning a value to an entry that did not previously exist, then the array is expanded as needed to contain all entries from the zero entry through the entry being assigned an initial value.

If a script expands an array using a SET statement and any lower numbered entries do not already exist, then by default, the SET statement propagates the source value into all lower numbered entries in the array that did not previously exist.

For example, if an array is uninitialized because a value was never assigned to any entry in the array, then the following SET statement propagates the specified source value into entries 0 through 10 of the target variable field array.

```
SET &target-variable(10) to 'initial value'
```

It is convenient to use the propagation feature of the SET statement to initialize an entire array to some desired initial value when the number of entries needed can be predicted. If propagation is not desired, for example, when the number of entries in an array is unpredictable and entries in the array are assigned initial values in other than ascending entry number order, then the SET statement ONLY keyword should be used to suppress the propagation feature. When the ONLY keyword is specified and lower numbered entries in an array must be created; then the lower numbered entries are assigned an initial zero-length null value and the specified source value is assigned only to the array entry specified in the SET statement.

The Script Language DROP statement deletes a variable field or an entire variable field array from the CA Enterprise Tester variable pool. After dropping a variable field, an array may be reinitialized using propagation.

Screen Fields

While a script is in session with an online system, the screen of the virtual terminal used for a session may be modified by the script only by using a Script Language TYPE statement to key data into the active session screen. Other statements may examine or otherwise refer to the virtual terminal screen using standard syntax to describe the screen field. The following is a description of the syntax used to refer to virtual terminal screen fields in Script Language statements other than the TYPE statement.

The general format of a reference to a virtual terminal screen field is:

```
< row column length >
```

where:

- < and > are symbols that identify a reference to a virtual terminal screen; a description of the screen field is enclosed within the screen field symbols.
- row is an unsigned integer or a variable field that contains the referenced screen row.
- column is an unsigned integer or a variable field that contains the referenced screen column.
- length is an unsigned integer or a variable field that contains the length of the referenced screen field. Length is optional. If not specified, the length is measured from the specified row and column and includes all screen data up to but not including the next screen field attribute. When a screen field is compared to another data item, an asterisk, * may be coded in the length field to specify that the length of the screen field is the same as the length of the other data item.

  For example:

  ```
  if <1 27 *> = 'ISPF/PDF PRIMARY OPTION MENU'
     go to primary-panel-now-displayed
  ```

  A reference to a screen field normally refers to the active session screen. CA Enterprise Tester allows a script to read screen images from a previously created CA Enterprise Tester trace data set using the Script Language TRACK statement. A script may specify the keyword, TRACKED, immediately preceding a standard screen field expression, to refer to a screen image obtained by the TRACK statement.

  For example:

  ```
  if tracked <1 27 *> = 'ISPF/PDF PRIMARY OPTION MENU'
     go to primary-panel-was-just-read-by-TRACK-statement
  ```

  The Script Language TYPE statement and a general description of a CA Enterprise Tester virtual terminal screen and keyboard is provided in Terminal Operation on page 5-29.

Session Control

A CA Enterprise Tester script starts a session between a CA Enterprise Tester virtual terminal and an online system using a Script Language LOGON statement. The statement specifies the name of the online system with which a session is desired.

For example:

```
LOGON CICS1
```

This LOGON statement assigns a CA Enterprise Tester virtual terminal and starts a session between the terminal and an online system named CICS1.

The names of the online systems available in a given network are assigned by system programmers during VTAM network definition. The name that you normally use to start a session with a system may be an alias of the name defined to VTAM. However, assuming that CA Enterprise Tester was appropriately customized during installation, you may use the online system name that you normally use to start a session. The actual name of an online system that has been defined to VTAM may also be used.

The LOGON statement may specify other parameters. For example:

```
LOGON TSO data 'user01 proc(uproc01)'
  24x80 43x80 extds
```

- The DATA keyword identifies data to be provided to the online system with the session request.
- The 24X80and 43X80 keywords specify the default and alternate screen sizes to be supported by the virtual terminal assigned to the session. 32X80 and 27X132 keywords identifying other screen sizes are also supported.
- The EXTDS keyword specifies that the virtual terminal is to support the 3270 extended data stream. An EXTDS terminal uses an extra buffer to hold extended color and highlighting attributes. A BASICDS terminal may be requested that does not support color and highlighting. A BASICDS terminal stores all field attributes and data in a single screen buffer.
- A LOGON statement may specify the VTAM mode table entry to be used to start a session.

  ```
  LOGON TSO MODEL MOD2
  ```

  The MODEL MOD2specification requests a mode table entry that has been set up for a 3270 model 2 terminal. Symbolic mode table names MOD3, MOD4 and MOD5 are also supported. When a symbolic mode table name is specified, the EXTDS or BASICDS keywords are also used in selecting the actual mode table entry to be used. The MODEL keyword may also be used to specify the actual name of a mode table entry that is in the mode table that is associated with CA Enterprise Tester virtual terminals.

- A LOGON statement may request a particular virtual terminal.

  ```
  LOGON TSO FROM PT99
  ```

The FROM keyword requests a terminal called PT99. Only names that have been defined to VTAM may be used. If a terminal is not specified, an inactive virtual terminal is assigned automatically.

Standard terminal session protocols prioritize terminal input messages over messages sent by an online system. If a terminal user could enter transactions as fast as a CA Enterprise Tester script, and the user kept entering transactions, an online system would never be able to send an online system initiated message to a terminal. When an online system needs to send a message that is not a response to a terminal input, a script must give the online system a chance to send the message.

When a session has just been started, online systems often send the first message on a session, to ask the terminal user to enter a user id and password. After starting a session, a CA Enterprise Tester script should give an online system a chance to send the initial sign on message. This may be accomplished using the Script Language INVITE statement. For example:

```
INVITE 1000
```

The INVITE statement gives an online system a specified amount of time to send a message. If a message is sent within the time limit, the INVITE statement ends when the message is displayed on the virtual terminal screen. Time is specified in CA Enterprise Tester Script Language statements in the format, hhmmssth (hours, minutes, seconds and tenths and hundredths of a second). The previous example allows 10 seconds.

- When a 3270 extended data stream session, (EXTDS), is started, it may be necessary to execute two INVITE statements at the start of a session. When a session is started, an online system may send a query to the terminal to inquire about supported screen sizes and other extended data stream capabilities. From the perspective of session message flow, a query looks much like any other message. Two INVITE statements may be required to receive: (1), the online system query; and (2), the initial sign on panel.

  There is an easy way to determine how many INVITE statements to code. Using the Interactive System, start a recording, then start a session with the online system using the SESSION command. The SESSION command may specify the EXTDS or BASICDS keyword. After signing on, end the session and the recording. The recording will contain the INVITE statements needed for a session with that online system.

After a session has been started, the Script Language TYPE statement, described in the next section of this chapter, may be used to key in sign on data, to enter subsequent transactions and to receive the online system response to those transactions. INVITE statements are not normally required after an online system has displayed the initial sign on panel.

Script Language DELAY statements may be automatically included in a script created with the CA Enterprise Tester Record Facility. The DELAY statement represents the amount of time elapsed since the last message was received from the online system, and the next AID key was entered by the terminal user, or actual user think time. When a session is reproduced by playing the script, the speed at which the script is played will closely resemble the speed of the actual interactive session. The speed may be varied at execution time by specifying the PERCENT keyword and a factor amount.

A CA Enterprise Tester script may end a session using a TYPE statement that enters a logoff transaction. A logoff transaction is any transaction that you enter to end a session with an online system. A script may also use the Script Language LOGOFF statement to end a session.

When a script is executed in a batch job, a session is automatically ended when a Script Language END or ABEND statement is executed.

If a Script Language LOGON statement is executed while a session is active, the session is ended prior to starting a new session.

CA Enterprise Tester scripts and Interactive System terminal users can share CA Enterprise Tester sessions.

- A terminal user may play a script that uses LOGON, INVITE and TYPE statements to start a session, sign on, and display an application menu. When a script that is played interactively ends without ending a session, the session remains active for manual use by the terminal user.
- A terminal user may start a session using the Interactive System SESSION Command. While the session is active, the user may play a script on the Interactive System port on which the session is active. The script can use the manually started session to enter any online transaction available on the system with Script Language TYPE statements.

Terminal Operation

A CA Enterprise Tester virtual terminal is available for use only while it is in session with an online system. The LOGON statement, described in the previous section of this chapter, may be used to start a session.

After a session has been started, a CA Enterprise Tester script operates a virtual terminal using the Script Language TYPE statement. The TYPE statement specifies exactly what is to be keyed into the virtual terminal. The virtual terminal is equipped with a full-function keyboard. The following keys are supported:

- Dual-case alphabetic characters, numbers and special symbols;
- An Erase EOF key;
- Cursor controls including Home, Tab, Backtab, Newline Cursor Select and Deselect and specific row and column cursor positioning;
- Clear, Enter, PA1-PA3 and PF1-PF24 keys;
- An ATTN key;
- A 3270 light pen.

  The TYPE statement processes a list of arguments that specify what data or other keys are to be pressed. Each simulated keystroke affects a virtual terminal in the same way that a real keystroke affects a real 3270 terminal. For example, the statement:

  ```
  TYPE 'CEMT' ENTER
  ```

  instructs CA Enterprise Tester to press each of the keys C, E, M and T, and then the Enter key. Each typed character is placed in the virtual terminal screen at the current cursor location, after which the cursor is advanced as it would be on a real 3270 terminal. When the Enter key is pressed, the data that has been typed is sent to the connected online system and script execution is suspended until a response has been received.

The Terminal Keyboard

The following paragraphs describe the operation of each key on the virtual terminal keyboard and describes how to express each keys using TYPE statement syntax and keywords:

- Data Keys

  Data keystrokes are placed in the virtual terminal screen at the current cursor location. An error results if an attempt is made to key data into a protected screen location. When a character of data has been keyed the cursor is advanced considering automatic skip and other field attributes present in the virtual terminal screen. When data is keyed into an unprotected screen field, the modified data tag in the attribute that controls the field is automatically turned on.

  Data keystrokes may be specified in a TYPE statement as a literal value that is enclosed in either single or double quotes. For example, TYPE 'ABCD', causes the string of characters ABCD to be keyed into the virtual terminal screen at the current cursor location, which is advanced after each keystroke.

Data may also be keyed from a variable field. The data contained in a variable field is keyed in exactly the same manner as literal data.

- Erase EOF Key

  The Erase EOF key causes the remainder of the current unprotected field beginning at the current cursor location to be cleared to nulls, without cursor movement.

  The Erase EOF key is represented in a list of TYPE statement arguments by the keyword, ERASE. The statement, TYPE 'ABC' ERASE, causes the character string ABC to be keyed in at the current cursor location after which the remainder of the current screen field, beginning at the screen position following the character C, is erased.

- Cursor Positioning Keys

  The Home, Tab, Backtab and Newline keys simulate the operation of the named hardware keys by appropriately repositioning the cursor. A virtual terminal cursor positioning key is pressed when the HOME, TAB, BACKTAB or NEWLINE keyword is encountered in a string of TYPE statement arguments.

  For example, the statement TYPE HOME 'ABC' TAB 'DEF', first positions the cursor at the first unprotected virtual terminal screen location, (HOME), then types the string of characters ABC while advancing the cursor appropriately, after which the cursor is moved to the next unprotected screen field, (TAB), before keying the string of characters DEF.

  The cursor may also be moved to a specified screen row and column. In a list of TYPE statement arguments, an argument that begins with a less-than symbol, < , is recognized as a cursor positioning order. The complete cursor positioning order encloses the screen row and column between less-than and greater-than symbols < and > , which are used like parentheses. The statement, TYPE <5,1> 'ABC' causes the cursor to be positioned at virtual terminal screen row 5 and screen column 1, at which the string of characters ABC is typed.

  Variable fields may be used to specify the screen row and column. The statement, TYPE <&R,&C> 'ABC', causes the cursor to be positioned at the row number that is contained in the variable field &Rand the column number that is contained in the variable field &C.

  Note: The Interactive System SPOT command provides an easy way to obtain the row and column of a screen field. See SPOT on page 3-38 for details.)

  TYPE statement keywords, SELECT and DESELECT perform the same operations on pen detectable selection or attention fields that are performed by 3270 light pen and cursor select keys.

- Attention Identifier Keys

When a Clear, Enter, PA1-PA3 or PF1-PF24 attention identifier (AID) key is typed, or a 3270 light pen is used on a light pen detectable field, the contents of the virtual terminal screen are sent to the online system for processing. The actual data sent simulates the 3270 short-read or read-modified command according to the AID key specified. After the data has been sent, script execution is suspended until a response has been received from the online system.

Note: TIMEOUT and HANDLE TIMEOUT statements may cause a TYPE statement to complete prior to receiving a response.)

Note: A short-read sends only an AID key value and is used for Clear and PA1-PA3 AID keys.

Note: A read-modified sends the AID key value, cursor position, and data from modified fields for Enter and PF1-PF24 AID keys.

In a list of TYPE statement arguments, the AID key is specified as one of the keywords, ENTER, CLEAR, PF1 through PF24, PA1 through PA3, or PEN. The statement, TYPE CLEAR causes the virtual terminal screen to be cleared to nulls after which the Clear AID is sent to the online system.

- ATTN Key

  When the Attention key, usually labeled ATTN on a real keyboard, is pressed, an Attention key signal is sent to an online system. The Attention key differs from other AID keys, in that it may be pressed even when the keyboard is inhibited while waiting on a response from an online system. Data is not sent to an online system when the Attention key is pressed.

  The statement, TYPE ATTN, presses the Attention key on the virtual terminal keyboard. When a TYPE statement specifies the ATTN keyword, the statement should specify no other TYPE statement arguments.

In a TYPE statement, literal data keystrokes, variable fields containing data to be typed, special keys and cursor positioning orders may be listed in any desired order or combination.

Consider the following example. An online transaction that updates records in an online data set, displays and gets input from a map that is displayed on the terminal screen (real or virtual). The map formats the following screen fields:

- At screen row 3, column 9, a request code field allows entry of a D to denote display, or a U to denote update.
- At screen row 4, column 9, a ten digit account number field allows the key field of a particular record to be specified.
- At screen row 5 and column 20, an amount field allows the entry of some number related to the account.

If the amount field is to be zeroed for a particular account, where the account number is in position 7 of a record that has been read from a user data set into a variable field, &RECORD, then the following statement may be used to key in all of the required information and press the Enter key:

```
TYPE 'U' TAB &RECORD(,7,10) <5,20> '0' ERASE ENTER
```

This statement assumes the cursor is initially positioned at the request code field, at which the character U is typed. The statement then tabs to the account number field at which the account number is keyed in from position 7 of the variable field, &RECORD. After keying in the account number, the statement explicitly positions the cursor to the amount field at row 5, column 20, types in the character 0, and then erases the remainder of the field. The statement then presses the Enter key to send the input to the connected online system.

An AID keyword or the ATTN keyword delimits the list of TYPE statement arguments to be keyed. In some situation, it might be convenient to key in some data, but to defer pressing an AID key until some other data is subsequently typed. This may be accomplished by simply not specifying an AID keyword. If no AID keyword is found in a TYPE statement, then the TYPE statement ends after keying the specified data. Additional data may be keyed by subsequent TYPE statements. When data is keyed, the keyboard state is changed from free to busy, exactly like a real keyboard. When data is keyed without pressing an AID key, a subsequent TYPE statement should eventually press an AID key to send typed data and to reset the keyboard state.

A TYPE statement that ends with an AID keyword causes typed data to be sent to the connected online system. When the data is sent, CA Enterprise Tester normally waits for the response from the online system, and the TYPE statement completes when an online system has sent a response that frees the terminal keyboard. When a TYPE statement completes, script execution resumes at the next script statement. Statements that follow the TYPE statement may refer to the virtual terminal screen to examine the data displayed by the online system. HANDLE OUTBOUND, TIMEOUT and HANDLE TIMEOUT statements may cause a TYPE statement to complete prior to receiving an online system response that frees the keyboard.

- A short-read sends only an AID key value and is used for Clear and PA1-PA3 AID keys.
- A read-modified sends the AID key value, cursor position and data from modified fields for Enter and PF1-PF24 AID keys.

The Terminal Screen

A CA Enterprise Tester virtual terminal screen is buffer in memory that contains field attributes and data that was displayed by an online system or that was keyed into the screen using the TYPE statement. A virtual terminal has two screen sizes, a default and an alternate, that are established when a session is started. Screen sizes may be requested on a LOGON statement by specifying two of the keywords, 24X80, 32X80, 43X80 or 27X132. Alternatively, the LOGON statement MODEL options, MOD2, MOD3, MOD4and MOD5, may be used to specify the implied screen sizes. System defaults are used when sizes are not specified or implied.

While a session is active, the size of the virtual screen is either the default or alternate screen size established for the session. The current size is controlled by write commands sent by the online system. When instructed by the online system, the size of the virtual screen is changed to either the default or alternate screen size. A script can determine the active screen size by looking at the variable fields, PTEROWS, PTECOLS and PTESIZE, which contain the current number of screen rows and columns and the screen buffer size in bytes (rows x columns).

Like real terminal screens, CA Enterprise Tester virtual terminal screens contain attributes. Online system orders place field attributes on a screen at desired locations to create screen fields. Field attributes control many properties including protection and intensity, and whether the cursor skips over a field when a character is keyed into the previous screen location. One bit in a field attribute, the modified data tag (MDT), is set by a script when data is keyed into an unprotected field. The MDT identifies what to send when the Enter key is pressed.

In addition to the basic field attributes, there are many other attributes that are not supported by all terminals, including extended field color and highlighting and character attributes that allow properties of individual characters to be specified. Aside from the fact that color can not be displayed on a monochrome terminal, there is another reason that some terminals do not support extended field attributes and character attributes; when extended attributes are taken into account, there are more combinations of attribute values than can be stored in a single position of a screen buffer. Terminals that support extended field attributes and character attributes have an extended attribute buffer (EAB) that is used to hold the extended field and character properties. Each location in the basic screen buffer corresponds to a location in the EAB that contains the extended properties of the field attribute or character at the screen location.

CA Enterprise Tester virtual terminals may be used to emulate terminals that support only basic field attributes and terminals that support extended field attributes and character attributes. The attributes to be supported by a virtual terminal that is used for a session may be specified on a LOGON statement by coding one of the keywords BASICDS or EXTDS. A system default is used when neither BASICDS nor EXTDS are specified.

- When a LOGON statement specifies BASICDS, a virtual terminal is configured with a single screen buffer in which is used to store basic field attributes and displayed screen data. If an online system sends extended attributes, they are tolerated, but the properties are discarded. The actual value of attributes defined by 3270 architecture are character values. So that attributes can be distinguished from characters when stored in a basic screen buffer, basic field attributes are translated to values that are not characters. To be specific, the 32 possible basic field attributes are stored in the basic screen buffer as binary values 1 through 32.
- When a LOGON statement specifies EXTDS, a virtual terminal is configured with both a screen buffer and an extended attribute buffer. The screen buffer is used to store displayed screen data and basic field attributes. When EXTDS is used for a session, basic field attributes are not translated to other values, but rather have the character value defined by 3270 architecture. One bit in the EAB indicates if the corresponding bit is a character or a field attribute. Other bits in the EAB contain the extended color and highlighting attributes of a corresponding field attribute or display character. Another bit in the EAB indicates that a corresponding character is a Graphic Escape character.

A CA Enterprise Tester screen format trace captures all supported properties of a screen and can be used to see BASICDS and EXTDS buffers.

Analyzing the Contents of a Virtual Terminal Screen Image

A CA Enterprise Tester script can refer to both screen data and basic field attributes using standard CA Enterprise Tester screen field syntax, (ie., <r,c,l>). The 3270 attribute values used in either BASICDS or EXTDS screen buffers are documented in 3270 Attribute Values on page 14-2. However, because attributes have different values when using BASICDS or EXTDS , scripts that refer to attribute values become terminal-type dependent. While the bit values contained in the EAB are documented, it is impossible for a script to examine the attributes directly. A CA Enterprise Tester script can use the SET statement ATTRIBUTES adjective to examine all of the attributes of a screen location in a terminal type independent manner. For example:

```
SET &attr to ATTRIBUTES <1 2>
IF &attr CONTAINS word 'modifiable'+
then
   type <1 2> 'x'
else
   log 'row 1, column 2 is not a modifiable location'
endif
```

The SET statement with the ATTRIBUTES adjective puts a list of the names of the attributes that describe screen location, row 1, column 2, in a specified variable field. The IF statement with the CONTAINS keyword asks if a particular attribute is in the list. 3270 Attribute Values on page 14-2 documents the names of all possible screen attributes.

Structuring Script Code

CA Enterprise Tester scripts that are created using the recording feature (see Recording Session Scripts on page 2-38), contain TYPE statements that capture the keystrokes entered by a terminal user. While recorded scripts often work correctly without modification, they are capable of repeating only a rote sequence of keyboard operations that mimic the previous actions of a terminal user. The recording mechanism captures only the physical actions of a person and not the decision making process that occurred when the person decided what data to enter or the decision making process that occurred when the person looked at the online system response and decided what to do next. When a recorded script is executed, if an online system responds to entered data by displaying a different panel, possibly because online system data has changed, then subsequent TYPE statements contained in the recorded script are likely to fail or to enter data invalidly. Scripts that perform functions in a production environment need logic to verify that transactions respond as expected and to take corrective action when errors are found.

The CA Enterprise Tester Script Language provides verbs that may be used to examine virtual terminal screen fields and other data and to structure conditionally executed script routines.

- The IF statement examines data items, evaluates relationships and conditionally executes structured script routines depending on the results of the comparison(s). The conditions that are specified in an IF statement determine which of two alternative routines are to be executed.

  When stated conditions are true, a routine is executed that is identified by the keyword, THEN, and that is delimited by ELSE or ENDIF. The keyword NEXT may follow THEN to do nothing when conditions are true.

  When stated conditions are false, a routine is executed that is identified by the keyword, ELSE and that is delimited by an ENDIF. The keyword NEXT may follow ELSE to do nothing when conditions are false. ELSE is optional and need not be coded if no action is to be taken when conditions are false.

  For example:

```
if <1 27 *> = 'ISPF/PDF PRIMARY OPTION MENU'
then   log 'Primary menu is now displayed.'
else   log 'The following TYPE returns to the primary menu.'   type home
'return' erase enter
endif
```

- DO and ENDDO statements mark the beginning and end of a loop that is repeated WHILE or UNTIL specified conditions are true or until the loop is ended by other script statements.

  For example:

  ```
  set &retries to 0
  do until <1 2 *> = 'IKJ56700A ENTER USERID'   set &retries up 1
     if &retries > 10
     thenlog 'USERID prompt not displayed.  I give up.'abend 1
     endif
     invite 1000enddotype &userid enter
  ```

  By default or if DO AFTER is specified, a DO routine is executed and the loop is repeated only after finding that specified conditions exist. If DO BEFORE is specified, a DO routine is executed before evaluating specified conditions and deciding if the loop is to be repeated.

  ITERATE and/or LEAVE instructions may be used in a DO loop to begin another iteration of the loop or to exit the loop immediately. DO

  FOREVER may be specified to repeat the loop until a LEAVE, EXIT, GOTO or other statement ends the loop. The LEAVE statement and IF-THEN-ELSE structures may be used in a DO loop that contains no conditions, to create select structures that are more flexible than conventional select structures.

  IF, DO WHILE and DO UNTIL statements support the comparison of screen fields, variable fields, literals and variables for any desired reason. Conditional statements compare one or more pairs of specified data items to determine if specified relationships are true or false. The relationship between a pair of data items may be expressed as one of the relational mnemonics, EQ, NE, GT, GE, LT, LE. Optionally, one or a logical pair of the relational symbols, >, <, =, ª (not), and \ (not), may be used to express a relationship.

  An additional condition, CONTAINS, may be used to search for a specified value on a virtual terminal screen or in a variable field. The CONTAINS condition is true when the argument being searched for is found in the specified screen or variable field. The search conducted when the CONTAINS condition is used is able to find a desired value even when the alphabetic case of the argument being sought differs from the case in the searched field. The search sees attributes that may exist between words on a screen, and binary zeros, as blanks, allowing screen search arguments to be based on what is seen on a screen. The CONTAINS condition also provides feedback in variable fields, that identifies the location of the argument found and the net length of the searched field that follows the argument found.

  ```
  if <1 1 80> CONTAINS 'ispf/pdf primary option menu'then   log 'the panel
  title begins in screen column ' &PTEARGCelse   log 'the panel title is not
  displayed on row 1'endif
  ```

  Refer to IF on page 6-55 or DO on page 6-25 for additional details on the CONTAINS condition.

Conditional statements may specify multiple conditions that are connected using AND and OR keywords. Conditions may be parenthesized in any desired way to control the order in which the conditions are evaluated.

The routine specified in a DO loop and the alternative routines specified in an IF statement may include any set of unlabeled Script Language statements, including additional IF and DO statements that may be nested to any desired number of levels.

In addition to the IF and DO statements that allow conditionally executed script code to be structured without using labels, the CA Enterprise Tester script language includes other verbs that may be used to execute labeled script routines, external script routines, programs that are written in other languages, as well as a list of one or more CA Enterprise Tester data items concatenated to construct a Script Language statement for immediate execution.

- A GOT statement unconditionally branches to another labeled script statement.

```
go to end-of-job
```

- A PERFORM statement goes to another labeled script statement, but when an EXIT statement is subsequently executed, script execution resumes at the statement that follows the PERFORM. Performed routines may be nested to any desired number of levels.

```
   perform subroutine
 .
 .
   end
subroutine
 .
 .
   exit
```

- The LINK statement loads and executes a named CA Enterprise Tester script, and completes when the script exits.

```
set July 8th, 1998 'today format &today english "weekday, month d, yyyy" '
link 'ptedate'
   if &date-error > ' '
then
log 'date error: ' &date-error
else
   log 'Today is ' &today
endif
```

- A CALL statement allows a script to call programs that are written in another language that supports standard z/OS linkage and parameter list conventions.

```
set &encode-parm(,1,8)  to 'ENCODE '
set &encode-parm(,9,8)  to 'myid  '
set &encode-parm(,19,8) to &my-password
```

```
call 'PTEIS' using &encode-parm
set &my-encoded-password to &encode-parm(,19,8)
```

- An EXECUTE statement specifies a list of one or more CA Enterprise Tester data items, including literals, variable fields, and other CA Enterprise Tester data items. The format of the EXECUTE statement is identical to the Script Language LOG statement.

  When an EXECUTE statement is processed, the values of the specified data items are concatenated to construct another Script Language statement. The constructed statement is then executed immediately.

  For example, if a variable field, &TARGET-FIELD, contains the value OPTION, then

  execution of the following statement:

  ```
  EXECUTE 'SET &' &TARGET-FIELD ' TO &SOURCE-FIELD'
  ```

  constructs a new Script Language statement by concatenating the values of the specified literal and variable data items to yield:

  ```
  SET &OPTION TO &SOURCE-FIELD
  ```

  The constructed statement is executed immediately.

Compatibility Form IF Statement

In addition to the structured IF-THEN-ELSE-ENDIF described previously, the CA Enterprise Tester Script Language supports an original form of the IF statement. Following a set of conditions, an IF statement may specify one of the verbs, GOTO, PERFORM, LINK or CALL, instead of THEN. When GOTO, PERFORM, LINK or CALL is specified, then the single statement is executed when the conditions are true, otherwise, script execution is resumed at the statement that follows the GOTO, PERFORM, LINK or CALL statement.

```
if &ptestate = 'OFFLINE'go to session-not-active
```

Variable Memory

Variable memory provides CA Enterprise Tester Scripts with general purpose working storage that may be used for any desired purpose.

For example:

- Numeric variables may be used as screen row and column indexes or screen field lengths in references to a virtual terminal screen.
- Numeric variables may be used as subscripts to reference occurrences in other variable arrays, or may be used to refer to substrings of variable fields.

- Records from user data sets may be read into or written from variable memory.
- Variables may be used as parameters to performed and linked script subroutines or as parameters to called programs that are written in other languages.
- The response from one transaction may be remembered in a variable field and later may be used as input to another transaction.
- CA Enterprise Tester defined variables provide operator replies, session status, and other specific information for examination by a script.

A variable field is a named field of information that is dynamically created when first referenced in a Script Language SET, READ or COMPUTE statement. A variable field may also be created via a JCL EXEC statement PARM specification, or the Interactive System Play Panel EXEC PARM data entry area.

A general description of CA Enterprise Tester variable memory and the syntax used in Script Language statements to refer to variable fields and variable field substrings is described in Variable Fields on page 5-21.

The Script Language SET statement provides numerous options that may be used to manipulate variable memory to accomplish a desired objective. Refer to SET on page 6-86 for precise descriptions of all SET statement options.

The following topics provide general descriptions and examples of some of the ways that character and numeric variables may be modified and used.

Character Variables

Character variables may contain data of any format and may be up to 32,760 characters in length. Variables that are created via JCL PARM specifications, those that are created via the READ statement, and those created by a SET statement that does not imply or specify that the variable is to be numeric, are considered to be character variables. The length of a character variable is normally determined to be the length of the data to which it is set. Substring notation may be used to explicitly control the length of a character variable field. The following examples illustrate ways to create character variable fields:

- Parameters may be passed to a script via the JCL statement PARM field. For example:

  ```
  //STEP1 EXEC PGM=PTEBS,PARM='SYSTEM=CICS1'
  ```

  The PARM specification SYSTEM=CICS1, causes the variable field, &SYSTEM to be set to the value CICS1. Variables that are created via the PARM field are established prior to the execution of the first script statement. Parameters may also be passed to scripts that are executed using the Interactive System via a Play Panel parameter field.

- The READ statement causes a record to be read from a data set identified by DD name, and to be placed into the specified variable field. The length of the variable field is the length of the record read.

  ```
  READ USERDD INTO &RECORD
  ```

- The SET statement may be used to set character variables to a literally specified value, to the value contained in another character variable field, to a substring of another character variable, or to a value obtained from a specified CA Enterprise Tester screen image field.

  ```
  SET &SWITCH  TO 'Y'
  SET &SWITCH  TO &OLDSW
  SET &FIELD   TO &RECORD(,20,5)
  SET &SCREENF TO <3,20,12>
  ```

- An occurrence of a variable array may be set or referenced by specifying an occurrence number in parentheses immediately following the variable field name. An occurrence number may be expressed as an unsigned integer, or a non-subscripted name of another variable field that contains the occurrence number.

  ```
  SET &DDNAME(1) TO 'USERDDN'SET &RESP(&N)  TO <24,1,80>LOG &RESP(&N)
  ```

  Variable arrays may be have more than one dimension. To refer to a variable field in a multi-dimension array, subscripts that denote the occurrence within each dimension are enclosed in a second set of parenthesis. The second set of parenthesis allow multiple subscripts to be distinguished from substring position and substring length specifications that may be coded following subscripts. For example:

  ```
  set &table( ( &i &j &k ) ) to &item
  ```

  An occurrence number may be any unsigned integer, including the value zero. A reference to occurrence zero refers to the same variable memory as a reference that does not specify an occurrence. The expressions &FIELD and &FIELD(0)refer to the same variable memory.

  When the value of an occurrence of a variable field is established with a SET statement, and some lower numbered occurrence has not been previously valued, then the SET statement will also establish the value for all lower numbered occurrences of the variable array to the value specified.

  For example, the following statement would set 24 occurrences of &FLAG, and &FLAG(0), to the value N:

  ```
  SET &FLAG(24) TO 'N'
  ```

  The propagation feature of the SET statement may be suppressed by specifying the ONLY keyword in the SET statement. Any lower-numbered entries in the array that have not been assigned a value previously will be initialized to a zero-length null value.

The Script Language DROP statement may be used to delete a variable field or an entire variable array. After an array has been dropped, it may be used again, even re-initialized using the propagation feature of the SET statement.

- A substring of a character variable field may be set or referenced by specifying a substring position and length following any specified subscripts, in the second and third positions within the parentheses that follow the variable field name. If the variable field being modified is not an entry in an array, then you may specify occurrence zero or just code a comma prior to the substring specifications.

  For example, to update a 5-byte field located at position 20 of a record stored in the variable field, &RECORD:

  ```
  SET &RECORD(,20,5) TO &NEWVAL
  ```

  Similar syntax is used to refer to a substring of a variable field in other Script Language statements. For example:

  ```
  IF &RECORD(,1,1) EQ '2' GOTO RECTYPE2
  ```

  If a substring of a variable field is set but the variable does not already exist, then the portion of the variable field being modified that precedes the specified substring, is initialized with blanks. If a substring position is specified, but a substring length is not, then the length of the substring to be set is determined to be the length of the data to which the substring is set. When the length of a substring being set is explicitly stated, but the data to which it is set is a different length, then the explicit length is used and the source data is truncated or padded with blanks to obtain the new substring value.

  For example, if the variable, &FIELD, has never been previously set and the following statement is executed:

  ```
  SET &FIELD(,2,4) TO 'SS'
  ```

  then &FIELD will be set to the value SS.

A SET statement may be used to concatenate two or more data items together to construct a target field value. The list of data items may include literals, screen fields and variable fields.

For example:

```
SET &X TO 'ROW ' &ROW '=' <&ROW,1,&PTECOLS>
```

The new target variable value is constructed by concatenating any literal data with the contents of screen and variable fields as specified.

The SET statement defines a number of adjectives that may be used to create variable values of a desired composition or format. These adjectives support pattern propagation, conversion of data to internal packed, binary, and zoned data formats, and the conversion of data between internal and external hexadecimal formats. A byte of data may also be converted to string of zeros and ones that reflect bit values or a string of zeros and ones may be converted to bits within a byte of data. SET statement adjectives also support the analysis of screen field attributes.

When subsequently referenced, variables that are the targets of these adjectives are treated as logical character values. The following examples illustrate the use of SET statement adjectives to create character variable fields.

- The ALL adjective may be used to propagate a source pattern through a target variable field.

  For example:

  ```
  SET &X(,1,132) TO ALL '-'
  ```

  causes the variable field, &X, to be filled with 132 hyphens.

- A variable may be set to PACKED, BINARY or ZONED data formats. When a SET statement specifies the keyword, TO, and one of these adjectives, a zoned decimal number is obtained from the source data specified and is converted to the designated internal data format. The source data may be expressed as a number, a literal value, or as a screen or variable field. In any case, the specified source data must contain an unsigned zoned decimal number. Any leading or trailing blanks or nulls found in the source field area ignored.

  For example:

  ```
  SET &X(,10,3) TO PACKED &COUNTER
  SET &X(,23,2) TO BINARY 1
  SET &X(,52,9) TO ZONED <&ROW,10,9>
  ```

  The length of the target PACKED, BINARY or ZONED number may be specified via substring notation. If a target substring length is specified, the source value is converted to internal format and is right justified in the target field. The target field is then left-filled with zeros of the designated internal format. If a substring length is not specified, then the length of the target field is determined according to the number of bytes required to contain the significant digits of the source number when expressed in the designated internal format. The high order bit of a binary target is considered to be a part of the number rather than a sign.

- When a variable field is set TO HEX, the source data must be a literal value, screen field or variable field that contains only the EBCDIC characters 0 through 9 and A through F. The source characters are combined two characters per byte to produce an internal hexadecimal representation of the character data.

  For example:

```
SET &X TO HEX 'FF'
```

If a source field contains an odd number of characters, then a leading zero is added to the source value before converting pairs of hexadecimal source characters to internal format. If the target variable specifies an explicit substring length that is other than half the number of source characters specified, then the source data is truncated or the target substring is padded with blanks.

- When a variable field is set FROM HEX, the source data is treated as internal hexadecimal data, and each byte is converted to two characters (0-9 and A-F) resulting in a displayable representation of the internal hexadecimal data.

  ```
  SET &X FROM HEX &RECORD(,5,4)
  ```

  If the target variable specifies an explicit substring length that is other than twice the number of source bytes specified, then the source data is truncated or the target substring is padded with blanks.

- The SET statement BITS adjective allows a script to examine and alter a single bit of data. For example:

  ```
  set &zeros-and-ones from bits &record(,10,1)
  if &zeros-and-ones(,1,1) = '1'   then set &zeros-and-ones(,1,1) to '0'   else
  set &zeros-and-ones(,1,1) to '1'
  endifset &record(,10,1) to bits &zeros-and-ones
  ```

- The SET statement CAPITAL adjective allows a script to convert all alphabetic characters in a field of data to upper or lowercase characters.

  For example:

  ```
  set &upper-case-title to capital <1 27 30>set &lower-case-title from capital
  <1 27 30>
  ```

- The SET statement IMAGE adjective supports the editing of screen data for display purposes. When the IMAGE adjective is used in a SET statement the source screen field is copied to a target variable field. During the copy operation attributes, data controlled by dark attributes and non-displayable data is replaced by blanks, and graphic escape characters are replaced with dashes, (-).

  For example:

  ```
  SET &PRTLINE TO IMAGE <24,1,80>
  ```

  copies the 24th line of a screen to a variable field, editing the target field for display purposes.

- The SET statement ATTRIBUTES adjective may be used to determine all of the attributes of a particular screen location. SET with ATTRIBUTES puts a list of the names of the attributes that describe a screen location in a specified variable field. A Script Language IF statement that uses the CONTAINS keyword may be used to check for any desired attribute.

  For example:

```
SET &attr to ATTRIBUTES <1 2>IF &attr CONTAINS word 'modifiable'then
   type <1 2> 'x'else
   log 'row 1, column 2 is not a modifiable location'endif
```

3270 Attribute Values on page 14-2 documents the names of all possible screen attributes.

Numeric Variables

A numeric variable used by the CA Enterprise Tester Script Language may be an unsigned integer or a decimal number. An edited decimal number is a character variable.

For a complete description of the use of numbers within a CA Enterprise Tester script, refer to Numbers on page 5-18.

While Script Language statements that use data numerically recognize edited decimal numbers in any variable field, in literals and in screen fields, numeric variables improve the performance of Script Language statements that use data numerically.

The value of a numeric variable begins with a minus sign if the number is negative, otherwise the value is unsigned. If a number has a significant integer, the numeric variable value contains only the significant integer digits which are adjacent to a minus sign if the value is negative. If a number has significant fraction digits or if a number is rounded by a COMPUTE statement to a particular number of decimal places, then the numeric variable value contains a decimal point followed by the fraction digits. A numeric variable value contains no imbedded blanks.

Because the length of a numeric variable field varies according to the magnitude and other properties of a number, it's length is unpredictable and references to a substring of a numeric variable field are illogical and are not permitted.

Script Language statements that compare data, (IF and DO), compare data either algebraically or as logical characters. Data is compared algebraically if either argument in a comparison is a decimal number or a numeric variable and if the other argument is an edited decimal number, otherwise the arguments are compared as logical characters.

A numeric variable field which is an unsigned integer may be used as a row or column index or as a screen field length when referring to a CA Enterprise Tester screen; or may be used as a subscript to refer to an occurrence of a variable array; or as a substring position or length when referring to a substring of a character variable field.

Although arrays of numeric variables are possible, only an unsubscripted numeric variable may be used as a subscript or substring position or length, when referring to an occurrence or substring of another variable field.

In addition to its use as an index, subscript, position or length, a numeric variable field may be used to count, may be used arithmetically, may be used as an intermediate field when converting data between different formats, may be used as a variable statement number in a GOTO statement, or may be used for any other purpose within a script where numeric values are required.

The following examples illustrate some ways to use numeric variables.

- A numeric variable is created when a variable field is SET to a number or to another variable that is numeric.

  In the following examples, &ROW and &SAVEROW become numeric variables:

  ```
  SET &ROW TO 1SET &SAVEROW TO &ROW
  ```

- A numeric variable field is created when a SET statement is used to obtain the length of a character variable field.

  For example, the statement:

  ```
  SET &RECLEN TO LENGTH &RECORD
  ```

  causes the numeric variable field, &RECLEN, to be SET to the current length of the character variable field, &RECORD. A SET statement that specifies the LENGTH adjective may refer to a variable field that has no value, in which case the target variable is set to zero (0). This technique may be used to determine if a variable was specified via the JCL EXEC statement PARM field, or through the Play Panel EXEC PARM data entry area.

- A numeric variable field is created when a SET statement uses an unqualified source value which is a number or numeric variable:

  ```
  set &julian-date to &record(,12,5)
  ```

- A numeric variable is created when a SET statement uses a source value which is described as FROM BINARY, PACKED, ZONED or DECIMAL; and a target substring position and length are not specified; and SIGN, PUNCTUATE and JUSTIFY are not used to edit the decimal result.

  For example, the following SET statements create decimal numbers:

  ```
  SET &DEC-NUMBER FROM PACKED &RECORD(,10,3)SET &DEC-NUMBER FROM BINARY
  &RECORD(,13,2)SET &DEC-NUMBER FROM ZONED <&ROW,20,10>
  ```

- ZONED and PACKED data up to 30 significant digits may be converted to and from decimal format.

  ```
  SET &RECORD(,32,12) TO PACKED PRECISION 2 &DEC-NUMBER
  ```

- BINARY numbers from one to eight bytes in length may be converted to and from decimal format.

```
SET &RECORD(,32,4) TO SIGNED BINARY &DEC-NUMBER
```

- Once a numeric variable is created, it may be set Up on DOWN by a constant or variable number.

  For example:

  ```
  SET &ROW UP 2SET &COUNT DOWN 1
  SET &TOTAL TO 0SET &COUNT FROM ZONED <&R,56,7>SET &TOTAL UP &COUNT
  ```

- A COMPUTE statement evaluates complex algebraic expressions and stores the result in a numeric variable.

  For example:

  ```
  COMPUTE &RESULT = (&A + <&row &col 10>) * (&B / .35 ROUND 2)
  ```

CA Enterprise Tester Defined Variables

A variable name which begins with the characters, PTE, is reserved. The content and use of these key elements is defined by CA Enterprise Tester. CA Enterprise Tester defined variables may not be modified by a script using a SET statement, but may otherwise be referenced wherever user-defined variables may be referenced. CA Enterprise Tester defined variables are used by CA Enterprise Tester to provide information to a script. When referred to in a Script Language statement, CA Enterprise Tester defined variables and user defined variables are prefixed by an ampersand, &, to distinguish variable references from other statement elements.

A list of all CA Enterprise Tester defined variables including the purpose and use is provided in PSL Variables on page 7-4.

Data Set Access

The CA Enterprise Tester Script Language provides data set access statements that allow a script to process fixed and variable length sequential data sets, VSAM Key Sequenced Data Sets, (KSDS), members of partitioned data sets, (PDS), and members of a generation data group, (GDG). Scripts may process other types of data organizations such as a data base by using the CALL statement to access subroutines written in languages which support standard z/OS call conventions.

Data set access is supported by a set of Script Language statements which include: ALLOCATE, OPEN, READ, WRITE, UPDATE, DELETE, CLOSE, and FREE statements.

Data set access statements refer to a data definition, (DD), statement that may be allocated to a script via JCL, the TSO ALLOC command, or the Script Language ALLOCATE statement.

- A script may read a fixed or variable length sequential data set, a member of a PDS or GDG, or a VSAM KSDS sequentially.

  The routine in the following example opens, and sequentially processes all of the records in a data set associated with the INFILE1 DD statement.

  ```
  OPEN INPUT INFILE1READ READ INFILE1 INTO &RECORD ENDFILE EOFSET &LENGTH TO
  LENGTH &RECORD*
  * process record now contained in &RECORD*GOTO READ
  EOF  CLOSE INFILE1END
  ```

  Each execution of the READ statement gets the next record from the data set and puts the record into the specified variable field, &RECORD. The script then puts the length of the record into the variable field, &LENGTH, before processing the record. When the end of the file is reached, the routine closes the data set.

  For a VSAM data set, the addition of the keyword, PREVIOUS, to the READ statement, would cause the script to read the data set backwards.

- A script may use the WRITE statement to create records in a fixed or variable length sequential data set, a member of a PDS or GDG, or add records to a VSAM KSDS.

  The script in the example below can be used to reproduce a sequential data set, to make a sequential copy of a VSAM KSDS, or to load an empty VSAM KSDS.

  ```
  OPEN INPUT  INFILE1 ENDFILE EOFOPEN OUTPUT OUTFILE
  READ READ  INFILE1 INTO &RECORDWRITE OUTFILE FROM &RECORDGOTO READ
  EOF  CLOSE INFILE1CLOSE OUTFILEEND
  ```

  If the OUTFILE data set in the example is an empty VSAM KSDS, then records are loaded sequentially, until a record is written out of key sequence, at which time records are inserted randomly.

  The WRITE statement may be used to construct the record to be written. The specifications listed following the WRITE statement FROM keyword may include a list of literals, variable fields and screen fields that are to be concatenated together to construct the record to be written.

  For example, if positions 1 through 20 of a variable field, &RECORD, contains the value CEDA INSTALL GR(XXX), and the screen field at row 23, column 2, contains the value NORMAL, then the WRITE statement:

  ```
  WRITE STATLOG FROM'TRAN=' &RECORD(,1,20) ' RESP=' <23,2,6>
  ```

  would construct and write the following record:

  ```
  "TRAN=CEDA INSTALL GR(XXX) RESP=NORMAL"
  ```

If necessary, the WRITE statement will pad a record with blanks or truncate it, in order to make the record conform to a data sets fixed length format or maximum record length.

- A script may read records in a VSAM KSDS directly.

  A VSAM record, identified by a full or generic key, may be read directly as follows:

  ```
  SET &KEY TO 'key-value'READ INFILE1 INTO &RECORD KEY &KEY
  ```

  The READ statement above would get the first record whose key is greater than or equal to key-value. NOTFOUND may be specified to request a record to be read only if a record matching the specified key exists:

  ```
  READ INFILE1 INTO &RECORD KEY &KEY NOTFOUND ERRLABEL
  ```

- A script may browse a VSAM KSDS, beginning at any point.

  While processing a VSAM KSDS, read position is remembered following each successful read. A sequential read gets the record that follows the record last read. When PREVIOUS is specified, a sequential read gets the record that precedes the record last read. Thus, a VSAM KSDS may be browsed sequentially, beginning at any point. The browse in the following example, sequentially reads all of the records in the data set whose key begins with ABC:

  ```
  SET &COUNT TO 0OPEN INPUT VSAMDD ENDFILE ENDBREAD VSAMDD INTO &RECORD KEY
  'ABC' NOTFOUND ENDB
  NEXT SET &COUNT UP 1READ VSAMDD INTO &RECORD NEXT NOTFOUND ENDBGOTO NEXT
  ENDB LOG &COUNT " records found with key 'ABC'"CLOSE VSAMDD
  ```

- The WRITE statement may be used to randomly insert records into an existing VSAM KSDS. The WRITE statement can be used only to add a record whose key is not already in the data set.

- The UPDATE statement may be used to replace a record in a VSAM KSDS that has a matching key.

  In the following example, records from an input data set are inserted into an existing VSAM KSDS. If the record key already exists in the data set, then the record is replaced.

  ```
  OPEN INPUT  INFILE1 ENDFILE EOFOPEN OUTPUT OUTFILE
  READ READ   INFILE1 INTO &RECORDWRITE  OUTFILE FROM &RECORD DUPKEY REPLGOTO
  READ
  REPL UPDATE OUTFILE FROM &RECORDGOTO READ
  EOF  CLOSE  INFILE1CLOSE  OUTFILEEND
  ```

- The DELETE statement may be used to delete one or more records from a VSAM KSDS, that are identified by a full or generic key.

  For following script deletes all records whose key begins with the value ABC.

  ```
  OPEN OUTPUT VSAMDSSET &KEY TO 'ABC'DELETE VSAMDS KEY &KEY NOTFOUND MESG
  MESG LOG &PTEDEL ' records deleted'CLOSE VSAMDS
  ```

- The FREE statement may be used to deallocate a data set associated with a DD name.

Processing Special Conditions and Errors

CA Enterprise Tester Script Language data set access statements support handling of special conditions, including end-of-file, record not found and duplicate keys, and also support handling of all logical and physical data set access errors, including VSAM errors and sequential data set abends.

- Data set access statements specify condition handling by coding named keyword conditions and the script routine that is to receive control if the condition is encountered. Script routines may be identified by statement label or statement number, by specifying a variable field that contains the statement label or number, or by coding an asterisk (*) to specify that script execution is to resume at the following statement.
  - The OPEN statement may specify an ERROR keyword to handle an open error and may specify a WARNING keyword to handle a VSAM open warning. An ERROR or ENDFILE keyword specified on an OPEN statement is remembered and will be used to handle conditions encountered by subsequent data set access statements that do not also specify handling for the condition encountered.
  - The READ statement may specify an ENDFILE keyword to specify handling for the ENDFILE condition, may specify a NOTFOUND keyword, or may specify how to handle a data set access error with the ERROR keyword.
  - The WRITE statement may specify how to handle a request to write a duplicate key with the DUPKEY keyword and may specify how to handle a data set access error with the ERROR keyword.
  - The UPDATE and DELETE statements may specify what to do if a specified record is not found with the NOTFOUND keyword or how to handle a data set access error with the ERROR keyword.
  - The ALLOCATE statement, the FREE statement, and the CLOSE statement may each specify how to handle an error condition with the ERROR keyword.

- If DUPKEY, NOTFOUND, ENDFILE, and WARNING conditions are not specifically handled with keyword identifiers, the conditions is considered to be an error and may be handled by a common error routine specified by the ERROR keyword. If handling for a condition is not specified and error handling is also not specified, then errors are handled according to the default or currently active HANDLE ILLOGIC specification.
- During the execution of every Script Language data set access statement, the variable field, PTERESP, is set to indicate the success of the request. An error handling routine may examine PTERESP to determine the exact cause of an error. Refer to PTERESP in CA Enterprise Tester Defined Variables on page 5-46 for a description of data set access response codes and particular error codes.

Message Logging

During script execution, CA Enterprise Tester produces a running log of messages and information that documents key events and errors as they are encountered. When executed as a step in a batch job, the CA Enterprise Tester log is written to the data set specified on the required PTELOG DD statement. When a CA Enterprise Tester script is executed using the Interactive System, the log is accessible via the Monitor Panel Message Log. When the Interactive System LOG command and the Log Panel are used to write logged messages to a data set, logged messages are written to both the Monitor Panel and the log data set.

If the LIST option is specified in the JCL or Play Panel EXEC PARM data entry area, CA Enterprise Tester logs the script as it is being loaded for execution. The FLOW option may be specified for debugging purposes to log Script Language statements each time they are executed. The RULOG option may be specified in a JCL PARM or a request unit log may be requested at the Interactive System Log Panel to log the 3270 data streams that flow on a session and messages that document RH indicators, RU names and logical unit states. See Batch JCL on page 5-6 for details.

A script may log additional messages in the CA Enterprise Tester log. This is accomplished using the Script Language LOG statement. The LOG statement specifies a list of data items that may include literals, variable fields or screen fields. The LOG statement concatenates the literals and the content of screen and variable fields to construct the message to be logged.

IF a field at row 5 and column 2 of the CA Enterprise Tester screen contains APMASTER and a variable field &STATUS contains the value A-OK, then the execution of the script statement:

```
LOG 'THE OPEN FOR ' <5,2> ' COMPLETED ' &STATUS
```

would result in the following log message:

```
hh.mm.ss.t - THE OPEN FOR APMASTER COMPLETED A-OK
```

Logged messages are automatically time stamped.

The following script subroutine may be invoked by the script statement, PERFORM LOGIT, to log selected screen images. The routine works for all screen sizes. The logged screen image is enclosed in a box that uses plus-signs for corners, and hyphens and vertical bars for the box top and bottom and box sides. The SET statement IMAGE adjective is used to convert screen attributes and data from dark screen fields to blanks, and graphic escape characters to hyphens (dashes).

```
logit  set &top(,1 132) to all '-'  log '+' &top(,1 &ptecols) '+'  set &row to 0
do while &row < &pterows
set &row up 1set &line to image < &row 1 &ptecols >log '|' &line '|'
  enddo
  log '+' &top(,1 &ptecols) '+'  exit
```

Operator Communication

The Script Language WTO and WTOR statements support one or two way communication between a CA Enterprise Tester script and a person. When executed in a batch job step, the WTO and WTOR statements write messages using z/OS WTO and WTOR macro statements, allowing messages to be routed to an operating system console or to the z/OS job log. The WTOR statement additionally stops script execution, waits for, and accepts an operator console response that is entered via the z/OS REPLY command.

When WTO and WTOR statements are executed by a script running under the control of the Interactive System, messages are logged in the Interactive System Monitor Panel Message Log. The WTOR statement additionally stops script execution, waits for, and accepts a user response that is entered via the Interactive System REPLY command.

The messages created by WTO and WTOR statements are constructed by concatenating a specified list of literals, variable fields and screen fields. By default, messages longer than 50 characters are broken into multiple messages that are no longer than 50 characters. The NOWRAP option of the WTO or WTOR statement may be used to specify that a single message is to be sent. The system WTO message length limit is 125 characters, and the WTOR message length limit is 121 characters.

WTO and WTOR statements support the specification of route and descriptor codes that allow statements executed in batch jobs to write messages to selected destinations and to specify the significance of a message according to z/OS WTO and WTOR macro statement rules. Refer to WTO on page 6-135 for details.

After sending the message to the z/OS console or to the Interactive System Log, the WTOR statement additionally solicits, waits for, and receives a reply from the console or Interactive System terminal user. The reply is placed in the variable field, PTEREPLY, for examination by the script.

For example, if a field at screen row 5 and column 2 contains APMASTER and a field at screen row 5 and column 64 contains OPEN FAILED then the following script statements, where SOLICIT is the routine label:

```
SOLICIT  WTO <5,2> ' OPEN RESPONSE WAS: ' <5,64>WTOR 'REPLY "RETRY", "IGNORE", OR
"ABEND"'
```

would result in the following messages:

```
APMASTER OPEN RESPONSE WAS: OPEN FAILEDREPLY "RETRY", "IGNORE", OR "ABEND"
```

After the second message is issued, the WTOR statement waits for the operator's reply. After putting the reply in PTEREPLY, script execution is resumed at the statement following the WTOR statement.

The following script statements examine the reply and proceed as specified:

```
IF &PTEREPLY = 'ABEND'  GO TO ABENDIF &PTEREPLY = 'IGNORE' GO TO BYPASSIF
&PTEREPLY = 'RETRY'  GO TO RETRYGO TO SOLICIT
```

In addition to the WTO and WTOR statements, the Script Language also provides a PAUSE statement that may be used to communicate with a CA Enterprise Tester Interactive System user. The PAUSE statement works only when a script is running under the control of the Interactive System, and is ignored when a script is played from a batch job. After logging information in the Monitor Panel Message Log, the PAUSE statement may be used to suspend script execution until it is resumed by the terminal user by entering an Interactive System VIEW, STEP or RESUME command.

Session Traces and Recordings

CA Enterprise Tester session trace and record features allow CA Enterprise Tester sessions to be captured in two distinctly different forms:

- The session trace feature captures screen images that represent terminal inputs and online system responses that occur on a traced CA Enterprise Tester session.
- The session recording feature generates a CA Enterprise Tester script that captures keystrokes and other session events in the form of Script Language LOGON, TYPE, INVITE and LOGOFF statements.

Any CA Enterprise Tester session may be traced or recorded. Interactive System facilities used to trace or record sessions that are started using the Interactive System SESSION command or that are driven by a script that is executed under Interactive System control, and general descriptions of the contents of session traces and recordings, are described in Tracing Session Screens on page 2-55 and Recording Session Scripts on page 2-38.

A CA Enterprise Tester session that is created by a CA Enterprise Tester script that is executed as a batch job may also be traced or recorded. The contents of a trace or recording of a batch session are identical to a trace or recording of an Interactive System session.

- A session trace may be automatically created when a PTETRACE DD statement is included in batch JCL. This feature may be controlled by the Script Language TRACE ON and TRACE OFF statement, the TRACE TO ddname statement, and the ENDTRACE statement.
- A session recording may be automatically created when a PTERECDD statement is included in batch JCL. This feature may be controlled by the Script Language RECORD ON and RECORD OFF statement, the RECORD TO ddname statement, and the ENDRECORD statement.

Batch JCL on page 5-6 describes PTETRACE and PTEREC data set considerations and requirements.

Script Language TRACE Statement

The Script Language TRACE statement may be used to control and enhance a trace of a session that is driven by a CA Enterprise Tester script.

- A TRACE statement may specify ON or OFF to selectively trace subsequent screen images.
- A TRACE TO ddname statement may specify the DD name to which the traced screen images are to be written.
- A TRACE statement may specify IMAGE, FORMAT or UPPERCASE to control the format and content of traced screen images. The format of a trace may also be controlled using the PARM field keywords, ITRACE, FTRACE or UTRACE.
- A TRACE statement may specify PACK or NOPACK to control whether data is written using the ISPF packed data format. The option should not be changed while a session is active. The packed data feature may also be controlled using the PARM field keywords, PTRACE, and NOPTRACE.
- A TRACE statement may create captions to be included on subsequently traced screen images.
- An ENDTRACE statement controls when a trace is ended.

The Script Language TRACE statement is fully documented in TRACE on page 6-112.

Script Language RECORD Statement

The Script Language RECORD statement may be used to control and enhance a recording of a session that is driven by a CA Enterprise Tester script.

- A RECORD statement may specify ON or OFF to selectively record Script Language statements that reflect subsequent session events.
- A RECORD TO ddname statement may specify the DD name to which the recording is to be written.
- A RECORD statement may specify PACK or NOPACK to control whether data is written using the ISPF packed data format. The option should not be changed while a session is active. The packed data feature may also be controlled using the PARM field keywords, PRECORD and NOPRECORD.
- A RECORD statement may write Script Language statements to a session recording that supplement those created by the session recording feature.
- An ENDRECORD statement controls when a recording is ended.

The Script Language RECORD statement is fully documented in RECORD on page 6-82.

Condition Handling and Diagnostics Aids

CA Enterprise Tester provides numerous features and controls that facilitate the control of session flow, error handling, problem determination and script debugging. This section summarizes controls and diagnostic aids that are supported by the CA Enterprise Tester Script Language. In addition to these features, the CA Enterprise Tester Interactive System provides interactive script monitoring and debugging capabilities. These features are presented in Script Monitoring and Interactive Debug on page 2-78.

Session flow control, error handling, problem determination and script debugging features supported by the CA Enterprise Tester Script Language include the following:

- The CA Enterprise Tester log contains messages that document significant session events as well as diagnostic messages describing syntax errors, illogical script requests and physical errors such as a lost session. The JCL PARM, RULOG, may be specified to request that raw 3270 data streams and detailed session flow messages be logged.
- Some critical messages may be sent to the z/OS Job Log, SYSLOG, or the operating system console.

- A HANDLE statement allows a script to control how unusual session events and session errors are handled.
- A LIMIT statement may be used to stop a runaway script at a user specified threshold.
- An ABEND statement may be used to terminate the current and subsequent steps of a batch job.
- Data set access statements support keywords that allow a script to deal with logical and physical data set access errors.
- A DUMP statement allows a script to display the current session screen and/or to display the current value of all script variables.

All CA Enterprise Tester messages whether sent to the console, job log, or CA Enterprise Tester Log, are prefixed by a unique message number. Messages are documented in CA Enterprise Tester Messages on page B-1.

Handling Unusual Session Events and Errors

By default, when script errors are detected or certain exceptional events occur while a script is executing, a message is issued and the script is terminated with a user 4095 abend. A Script Language HANDLE statement permits the script to specify the action to be taken by CA Enterprise Tester in the event of certain exceptional session events and script errors. HANDLE statement options are fully described in HANDLE in topic 6.25.

- A HANDLE CONTENTION statement specifies the action to be taken whenever a script TYPE statement and the connected online system both attempt to send a message at the same time. CONTENTION options are to reject the online system message, to receive and process the message then send the message being sent by the script, or to receive and process the online system message, but resume script processing at a specified script statement after processing the online system message.
- A HANDLE EVENTS statement may be used to process session events whenever they occur, regardless of where the script is executing. Options permit online system messages to be received and processed whenever they are detected. After processing an event automatically, script execution may be resumed without interruption or may be resumed at a specified script label.
- A HANDLE ILLOGIC statement specifies the action to be taken when a script logic problem is encountered. ILLOGIC options are to abnormally terminate the script, ignore the error and resume script execution at the next script statement, or to terminate execution of the statement and resume script execution at a specified script statement.

- A HANDLE LOGONFAIL statement specifies the action to be taken if a problem is encountered during the execution of a LOGON statement. LOGONFAIL options are to treat the LOGON failure as a session failure, automatically prompt the console or terminal operator for assistance in problem determination and correction, or to specify a script statement to be given control when a LOGON problem is encountered.
- A HANDLE OUTBOUND statement allows a script to examine online system messages that do not free the terminal keyboard. By default, only messages that free the terminal keyboard cause a TYPE or INVITE statement to complete. A script may request that TYPE and INVITE statements complete after receiving messages that do not free the terminal keyboard and may specify that script execution is to continue at the statement that follows the TYPE or INVITE statement or that script execution is to resume at a specified script label when a message is received that does not free the keyboard. After receiving a message that leaves a session in the outbound state, a script may use the INVITE statement to receive the next message.
- A HANDLE RECONNECT statement specifies the action to be taken whenever an active session is passed by an online system to another online system. RECONNECT options are to terminate the session and treat the condition as ILLOGIC, accept the new session and resume script execution at the next script statement, or to accept the new session and resume script execution at a specified script statement.
- A HANDLE RESPONSES statement provides options that may be used to process an unpredictable number of responses by an online system to terminal input from a recorded script. The statement options may be used to make a recorded script smarter without modifying the recorded script.
- A HANDLE SESSIONFAIL statement specifies the action to be taken if a session is abnormally lost during script execution. SESSIONFAIL options are to abnormally terminate the script, provide status information and continue script execution, or to specify a script statement to be given control in the event of an abnormal session loss.
- A HANDLE TIMEOUT statement and the TIMEOUT statement may be used to limit the time that CA Enterprise Tester will wait for the completion of an in-progress session event and to specify how to handle a time out condition. Options are to simply complete execution of a LOGON, TYPE or INVITE statement, and resume at the next script statement or to resume at a specified script label.

- A HANDLE TURNAROUND statement controls the operation of the Script Language TYPE statement. The TURNAROUND keyword refers to the point in a session that occurs when a TYPE statement has sent input to an online system and changes the session direction to outbound. By default, after changing the session direction to outbound, the TYPE statements waits for and receives the online system response before completing. When HANDLE TURNAROUNDOFF is specified, TYPE statements complete after changing the session direction to outbound, and do not wait for a response. The TYPE statement becomes a send-only operation. The INVITE statement is subsequently used to receive online system responses.

Controlling Script Loops

A Script Language LIMIT statement permits a script to control script loops. A LIMIT statement allows the script to specify the maximum number of times any script statement may be reasonably executed. If any script statement is executed more times than the LIMIT, then the script is abnormally terminated.

Forcing a Script ABEND

A Script Language ABEND statement may be executed to force abnormal termination of the executing script. When a script is being executed as a step in a batch job, the ABEND statement not only abnormally terminates the current job step but will also cause subsequent job steps to be flushed, depending on the EXEC statement CONDITION parameters specified for any subsequent steps in the job.

Requesting a Dump

When a script is terminated by an ABEND statement, when a script is terminated due to ILLOGIC, or when requested by executing a DUMP statement, then a formatted dump showing the content of all variable fields, and a dump of the virtual terminal screen is written to the CA Enterprise Tester Log.

Session Flows

The orderly exchange of messages between terminals and online systems is made possible by a set of rules that govern the flow of messages on a session. This set of rules is defined by System Network Architecture (SNA). SNA Type 2 Logical Unit protocols define the rules that govern sessions between 3270-compatible devices and online systems. By conforming to this set of rules, CA Enterprise Tester, like a 3270 terminal, is defined to be an SNA Type 2 Logical Unit.

Different online systems implement various optional features of the protocol in slightly different ways. A basic understanding of LU Type 2 protocol and the ways in which these protocols are implemented by various online systems will assist the author of CA Enterprise Tester scripts in creating applications that successfully accomplish any desired purpose.

The following sections describe the flow and control of messages on sessions between terminals and online systems, and Script Language features that allow a script to deal with the various ways in which these protocols are implemented by different online systems and applications.

Bracket Protocol

A session between a terminal and any online system consists of one or more units of work, each of which is called a bracket. A bracket consists of an initial message sent by one party in a session, to request a service of the other party, and any subsequent messages that are exchanged by the session partners while processing the initial request. Once a bracket has been initiated, each session partner agrees to send messages only in turn, and to wait for a message from the other party prior to sending another message.

While a terminal is sending messages to an online system, the session is in a bracket and has inbound direction. The online system will receive messages until an inbound message specifies change direction, at which time the session becomes outbound. While a bracket has outbound direction, the terminal keyboard is inhibited, and the terminal will receive messages until the online system changes session direction to inbound or until the online system ends the bracket. A session has inbound or outbound direction only while the session is in bracket.

Before a bracket is started or after a bracket has been ended, that is, while a session is between brackets, the session is in the contention state. While in the contention state, either party in the session, the terminal or the online system, may begin a bracket and send a message to the other party. A terminal begins a bracket when a user presses the Enter key or other attention identifier key. An online system may begin a bracket to send a notification or other message to a terminal. Bracket protocol rules dictate that only an online system may end a bracket.

- For example, when a user presses the Enter key on a terminal, a message is sent to the online system. The message will begin a bracket if one is not already active. If the message requests that some particular data be displayed at the terminal, the online system will send a message in response to the terminal request that contains the desired data, and may end the bracket.

- A bracket may include additional exchanges of information between the terminal and the online system. For example, while processing a terminal request, an online program might determine that additional information is required in order to respond properly. The program might respond to the terminal request with a message to solicit the needed information and wait for the next terminal input. The message would change the session direction to inbound and free the terminal keyboard, but not end the bracket. When the information is entered by the terminal user, a message is sent to the waiting online system and the session direction is changed to outbound. The online system might make yet another request, or might finally prepare the desired display and send it to the terminal, and end the bracket.

Thus, a bracket consists of a strictly controlled exchange of messages between session partners. While a bracket is active, each partner is either waiting on a message from the other, or is in a state in which it is required to send the next message on the session. While a bracket is active and session direction is outbound, the terminal keyboard remains inhibited. The terminal keyboard is free while the session is in the contention state that exists between brackets or while a bracket is active and session direction is inbound.

The following is a summary of the rules of bracket protocol:

- A session is in one of two states at all times, either contention or bracket.
- After session initiation and while a session is between brackets a session is in the contention state. While a session is in the contention state, either party in the session may begin a bracket and send a message.
- While a session is in the bracket state, the session has direction, either inbound or outbound, indicating whether the terminal or the online system may send messages. The sending session partner indicates when session direction is to be changed.
- If while in the contention state, both parties in a session attempt to start a bracket at the same time, the bracket begun by the terminal is given preference and the bracket started by the online system is rejected.
- Only an online system may end a bracket. When a bracket is ended, the session returns to the contention state.

Sessions with different online systems take on different personalities depending on the way in which these rules are implemented.

- CICS provides options that permit CICS applications to control the extent of a bracket. CICS ends a bracket only when a CICS application returns control to CICS.

- A CICS conversational application exchanges multiple messages with a terminal within a single bracket. After sending a message that frees the terminal keyboard and changes session direction to inbound, a conversational application waits on subsequent terminal input. No additional message will be sent to the terminal until the next terminal input is received. CEMT, CEDA and CECI are examples of CICS conversational applications.
- Most CICS applications operate in pseudo-conversational mode. After a pseudo-conversational application sends a message to a terminal, the application returns control to CICS, and specifies the CICS application that is to process the next terminal input. When an application returns control to CICS, CICS ends the current bracket. While waiting on terminal input, the session is in the contention state. Prior to receiving the next terminal input, it is possible for CICS to begin a bracket and to start a transaction that will send a message to the terminal.
- The CICS automatic transaction initiate (ATI) feature permits a CICS application to start a transaction on a session. When a transaction is started by the ATI mechanism, CICS begins a bracket that is used by the started transaction to send outbound messages. An electronic mail facility might use this feature to notify a terminal user of the availability of mail sent by another terminal user. Online applications often execute a EXEC CICS START TRANID(xxxx) TERMID(yyyy) command when a user enters a request to transfer to another CICS application. After the starting transaction ends, CICS uses ATI to start the transaction named in the START command.

- TSO treats an entire session as one large bracket. After sending a message to a terminal and changing direction, TSO waits until the terminal user presses the Enter key or other attention identifier key, prior to sending another message to the terminal. TSO rarely ends a bracket. Once the original bracket is started on a session with TSO, each session partner takes turns sending messages.
- NETVIEW treats each message exchanged by a terminal and itself as a bracket. When a message is received from a terminal, NETVIEW always responds with a message that frees the terminal keyboard and ends the bracket started by the terminal. This logic leaves the terminal free to enter new requests while prior requests are being processed. After a request has been processed, NETVIEW sends the response to the terminal as a separate bracket, freeing the keyboard and ending the bracket after each message is sent.

Flow Control

The CA Enterprise Tester Script Language provides features that permit a script to deal with any implementation of SNA Logical Unit Type 2 protocols. This section describes the relationship between the various Script Language statements and bracket protocol and features of the language that allow a script to analyze and control the flow of messages on a session.

A script may determine the state of a session by examining the contents of the virtual terminal screen or by examining CA Enterprise Tester defined variables that indicate the state of the session. CA Enterprise Tester also provides features that may be used by a script to automate or control the handling of the various session states, and to handle session errors and other special situations. The following is a discussion of techniques and features that may be employed within a script to create intelligent and reliable session dialogs.

If an online program can respond to a terminal input in a number of ways, then an intelligent CA Enterprise Tester script could examine the response contained in the virtual terminal screen buffer, using the Script Language IF statement, and control the execution of the script depending on messages sent by an online system. This technique can be used to check for messages that indicate the success or failure of a terminal request, and to act accordingly. For example:

```
IF <24,2,18> NE 'COMPLETED NORMALLY'   PERFORM ERROR
```

While a session is in the contention or inbound states, a CA Enterprise Tester Script may send a message to an online system by executing a Script Language TYPE statement. While a session is in the outbound state, the virtual terminal keyboard is inhibited and the TYPE statement may not be used to enter data or press enter. Depending on the online system with which a CA Enterprise Tester script is communicating and the session handling options specified by the script, it may be necessary for a script to analyze the state of a session.

The IF statement may also be used to examine CA Enterprise Tester defined variable fields that indicate the state of a session and other session related information. In particular, the variable field, PTESTATE, is used to inform a CA Enterprise Tester script of the current state of a session. The logic of a script may be controlled according to the indicated state. For example:

```
IF &PTESTATE EQ 'OUTBOUND'   GOTO INVITE
```

The PTESTATE variable normally contains one of the following values:

- OFFLINE indicates that a session is not active.

- ONLINE indicates that a CA Enterprise Tester session has just been started by the script, that a new session has been started by an online system by passing the session to another online system or that traffic on the session has been stopped and restarted by the online system by sending SNA CLEAR and SDT commands. When PTESTATE is ONLINE, the session is in the contention state.
- CONTENTION indicates that the previous bracket on a session was ended by the online system and that the session is in the contention state. While in the contention state, either session partner may send a message to the other. A script may send a message by executing a TYPE statement. A script may check for and receive a message sent by an online system by executing an INVITE statement.
- INBOUND indicates that a bracket is active on a session and that the session direction is inbound from the terminal to the online system. The next message on the session must be sent by the script by executing a TYPE statement. Session protocol ensures that the online system will not send a message until a message is sent by the script.
- OUTBOUND indicates that a bracket is active on a session and that the session direction is outbound from an online system to the terminal. Because TYPE and INVITE statements normally complete only when the keyboard is freed, a script does not normally see the OUTBOUND state. A script that uses HANDLE OUTBOUND, TIMEOUT or EVENTS statements may receive control in the OUTBOUND state.

  While in the OUTBOUND state, the TYPE statement may be used only to press the Attention key, and may not enter data or press the Enter key. The next normal message while a session is outbound must be sent by the online system. A script may receive the message by executing an INVITE statement.

Note: Many other session states are possible. However, other states are seen by a script only when CA Enterprise Tester TIMEOUT and HANDLE TIMEOUT statements are used to limit the time that CA Enterprise Tester will wait for the completion of an in-progress session event. CA Enterprise Tester Logical Unit States on page 14-7 documents other states that may be seen by a script if a session times out.

For security purposes, most online systems require that a user sign on before using online system applications. The sign on process is usually initiated by the online system immediately after accepting a session with a terminal, by sending a message that solicits the user's identification and password. An initial sign on request is an example of a bracket that is started by an online system. A script may anticipate and receive the sign on request by executing an INVITE statement. An INVITE statement may also be required at the start of an session that uses the 3270 extended data stream, to allow the online system to send a query to the terminal to determine terminal screen sizes and other extended data stream capabilities.

Online systems may send messages while in the contention state for other reasons. For example, an information retrieval application might be designed to accept terminal requests, then immediately end the bracket, freeing the user's keyboard so that additional request can be entered while searching for the requested information. At a later time, the online program might start a bracket and send a message to notify the terminal user that a reply to a previous request is now available for display. A script that communicates with an online system that starts brackets may use the INVITE statement to receive anticipated messages.

While in the contention state, if the terminal user has already begun to key data into a terminal, the terminal's controller will reject a message sent by an online system to the terminal. The feature protects the terminal screen so that keyed in data will not be overlayed by a message from the online system before the user has a chance to enter the data being keyed. The rejection of an online system message by a controller after a terminal user begins to key data enforces the protocol rule that gives precedence to terminal input when both a terminal and an online system attempt to send a message at the same time. The terminal controller considers a terminal initiated bracket to have begun when a terminal user begins to key data. An online system may deal with a rejected message by sending the message after input is received from the terminal.

By default, CA Enterprise Tester protects the virtual terminal screen in the same way that a terminal controller protects a real terminal screen. While processing a TYPE statement, if it is discovered that an unsolicited message was sent to the terminal by the online system, then CA Enterprise Tester will automatically reject the message prior to sending the data specified by the TYPE statement.

Application Initiated Brackets

If a CA Enterprise Tester Script invokes an online system or application that initiates brackets, then the script must implement CA Enterprise Tester features that check for and permit online system initiated brackets to be started on the session.

These features apply only while a session is in the contention state, that is, while PTESTATE is ONLINE or CONTENTION. The INVITE statement instructs CA Enterprise Tester to check for and receive any message that may have been sent by an online system. If the online system has sent a message, the message is received and is placed in the virtual terminal screen buffer. The CA Enterprise Tester defined variable, PTECC, is set to the count of messages received from the online system. The successful receipt of messages from an online system while a session is in the contention state is often a matter of timing. An INVITE statement may specify an interval of time that should be allowed for receipt of an anticipated message. Specification of a suitable interval will give the online system time to send the message. If an online system message is not waiting to be received when an INVITE statement is executed, or if the INVITE statement specified a time interval and no message is received within the interval, or if the online system passes the session to another online system or resets and restarts session traffic during INVITE processing then PTECC is set to 0 (zero) to indicate that a message was not received. If the execution of an INVITE statement results in a PTECC value of zero, but the CA Enterprise Tester script is logically assured that the online system will send a message, then it is likely that the INVITE statement did not specify an adequate INVITE interval. An INVITE statement will complete immediately when a message is received from an online system, regardless of the interval specified, so it is preferable to specify generous INVITE intervals.

If an online system passes sessions to other systems or an online system resets and restarts session traffic, it may be necessary to execute multiple INVITE statements to receive an anticipated message. When an INVITE statement completes and a message is not received, PTESTATE will be set to ONLINE if the session has been passed or if session traffic has been restarted.

A HANDLE CONTENTION statement may also be used to permit an online system to initiate a bracket. The statement allows a script to specify the action to be taken by the TYPE statement in the event that an online system initiated bracket is waiting to be started when a TYPE statement is executed.

- HANDLE CONTENTION OFF is the default CA Enterprise Tester specification for handling message conflicts while a session is in the contention state. During the execution of a TYPE statement, if it is discovered that the online system has attempted to start a bracket, and if contention handling is OFF, then the TYPE statement will automatically reject the online systems request before sending the terminal input specified in the TYPE statement.

- HANDLE CONTENTION ON specifies that when a TYPE statement is executed, that if a message sent by the online system is waiting to be received, then the message is to be received and placed in the CA Enterprise Tester screen, after which the TYPE operation will be resumed. If the TYPE statement specifies an AID key, a message will be sent to the online system. If the message sent by the online system has altered the format of the CA Enterprise Tester screen, then it may be illogical to resume the TYPE operation.
- HANDLE CONTENTION AT label also specifies that the TYPE statement should give priority to a bracket initiated by the online system, but that if an online system message is received and processed prior to performing the TYPE operation, that the statement should be terminated prior to keying any of the specified data. Instead, the variable field, PTELINE, is set to the line number of the TYPE statement at which the contention condition was encountered and script execution is resumed at the script label specified in the HANDLE CONTENTION statement. The specified routine may analyze the received message and take any appropriate course of action. If execution of the TYPE statement at which the contention condition was encountered is the appropriate action, then the condition handling routine may GOTO &PTELINE to retry the TYPE operation.

When a session's direction is outbound, an online system may send any number of outbound messages to a terminal without changing the session direction to inbound. A message sent to a terminal that does not change direction may be sent to inform a terminal user of the progress being made by a particularly long request, or might be used by an online system to perform piecemeal construction of a complex terminal display. By default, a CA Enterprise Tester script does not see individual outbound messages that leave the session direction outbound. Rather, CA Enterprise Tester will receive all outbound messages until a message is received that changes the session direction to inbound or that ends the current bracket. A HANDLE OUTBOUND statement may be executed by a script to specify how CA Enterprise Tester is to deal with the receipt of outbound messages that do not end a bracket and that do not change the session's direction to inbound.

- HANDLE OUTBOUND ON, the default specification for handling outbound messages that do not change session direction to inbound, specifies that the TYPE or INVITE statement is to continue receiving outbound messages until a message is received that changes session direction to inbound or until a message is received that ends the current bracket. By default, or if a HANDLE OUTBOUND ON statement is executed by a script, then the script will always be in a position to send messages and the TYPE statement will always be executable, provided that a session is active. The variable, PTESTATE will never contain the value OUTBOUND.

- HANDLE OUTBOUND OFF specifies that when an outbound message that does not change direction or end a bracket, is received during the execution of a TYPE or INVITE statement, that the statement is to complete normally after setting PTESTATE to the value OUTBOUND. A script that receives messages that leave session direction outbound, may examine the message, but may not modify the virtual terminal screen by executing a TYPE statement. An INVITE statement must be executed to receive the next outbound message.
- HANDLE OUTBOUND AT label specifies that when an outbound message that does not change direction or end a bracket, is received during the execution of a TYPE or INVITE statement, that PTESTATE is to be set to the value OUTBOUND, that PTELINE is to be set to the line number of the statement being executed, and that script execution is to be resumed at the specified script label. After examining the message, the script must execute an INVITE statement to receive the next outbound message.

Passed and Queued Sessions

SNA protocols and VTAM support session management features that allow online systems to manipulate terminal sessions:

- An online system may end a session with a terminal, specifying that the terminal is to automatically accept a forthcoming session with another named online system. This document will refer to the new session as a passed session.
- An online system may request a session with a terminal that is already in session with another online system, and may specify that the session is to be queued so that when the active session ends, the terminal will automatically accept the queued session request. This document will refer to these serially accepted sessions as queued sessions.

Some online systems, notably TSO and NETVIEW, implement passed sessions as a part of normal session initiation.

For example, when a user logs onto TSO, the application id specified by the user might be TSO. During the logon process, TSO creates a private address space for the user and assigns a different VTAM application id to be used for the session between the terminal and the user's private address space. The application id assigned might be a name like TSO10102. Just after accepting the requested session with TSO, TSO sends a command to the terminal that terminates the session and instructs the terminal to accept a session with TSO10102. This sequence of events is completely transparent to a terminal user.

Online systems may pass sessions for other reasons. An online system may provide a transaction that allows a user to request transfer of a session directly to another online system.

An online system may implement passed and queued sessions with a terminal to allow a number of online systems to communicate serially with the terminal. For example, if a terminal user selects an online system service from a menu, and the service is actually provided by another online system, the online system may pass the terminal session to the other online system, then queue another session request to the terminal so that when the other online system ends the session, the terminal will accept a new session with the original online system. The IBM Information Network implements passed and queued sessions to allow transparent access to network services provided by a number of online systems, where the online systems operate on different computers running at different physical installations.

If a CA Enterprise Tester script initiates a session with TSO or another online system that implements passed or queued sessions, then it may be necessary for the script to recognize and deal with the session events. The CA Enterprise Tester Batch System recognizes a new session during the execution of a TYPE or INVITE statement. When a session is passed, any bracket that was active on the session is terminated, the CA Enterprise Tester screen is cleared to nulls, and the variable field PTESTATE is set to ONLINE to indicate that a new session has been established.

The HANDLE RECONNECT statement allows a script to specify the action to be taken if a new session has been established during the execution of a TYPE or INVITE statement.

- HANDLE RECONNECT ON, the default specification for handling of passed and queued sessions, specifies that new sessions are acceptable. When a session is passed, CA Enterprise Tester will update the variable field, PTEPLU, to provide the new VTAM application id to the script, and will set PTESTATE to ONLINE to indicate that a new session has been established. A script may examine PTESTATE using an IF statement to detect the acceptance of a passed or queued session.
- HANDLE RECONNECT AT label specifies the name of the script statement at which script execution is to be resumed in the event that a new session is established. During the execution of a TYPE or INVITE statement, when a passed or queued session is accepted, the variable field, PTEPLU, is updated to contain the new online system's application ID, the variable field, PTELINE, is set to the line number of the TYPE or INVITE statement that was being executed when the new session was accepted, and script execution is resumed at the script label specified in the HANDLE statement. The script's RECONNECT handling routine may retry the TYPE or INVITE statement by executing the statement, GOTO &PTELINE.
- A HANDLE RECONNECT OFF statement specifies that passed and queued sessions are not acceptable. When a passed or queued session is detected, the new session is immediately terminated and the CA Enterprise Tester ILLOGIC condition is raised.

A session may be passed by an online system during the execution of a LOGON statement. This will happen if the online system passes the session prior to sending a signal to start data traffic on the session. When a session is passed during LOGON processing, the flow of messages on the session is not interrupted. If an online system passes a session during execution of a LOGON statement, CA Enterprise Tester accepts the new session and sets PTEPLU to the new online system application id. The HANDLE RECONNECT statement does not apply to sessions passed during LOGON processing. NETVIEW is an example of an online system that will pass a session during the LOGON process.

Chapter 6: PSL Reference

This chapter documents the syntax of and options supported by all CA Enterprise Tester Script Language verbs, arranged alphabetically by verb name.

- PSL Reference
- ABEND
- ALLOCATE
- AND
- CALL
- CLOSE
- COMPUTE
- DATE
- DELAY
- DELETE
- DO
- DROP
- DUMP
- ELSE
- ENCODE
- END
- ENDDO
- ENDIF
- ENDRECORD
- ENDTRACE
- EXECUTE
- EXIT
- FLOW
- FREE
- GOTO
- HANDLE
- IF
- INVITE

- LIMIT
- LINK
- LOG
- LOGOFF
- LOGON
- MASK
- NEXT
- OPEN
- OR
- PAUSE
- PERFORM
- READ
- RECORD
- SET
- THEN
- THINK
- TIME
- TIMEOUT
- TRACE
- TRACK
- TYPE
- UPDATE
- WAIT
- WRITE
- WTO
- WTOR

PSL Reference

This section documents the syntax and specifications of all CA Enterprise Tester Script Language statements. Statement descriptions are arranged alphabetically according to verb name.

SYNTAX NOTATION FORMAT

- CA Enterprise Tester Script Language verbs and keywords are in UPPERCASE TEXT
- User data is listed in lowercase bold text and can be literal data or variables
- Optional parameters are enclosed in parentheses, ( )
- Alternative keywords and/or parameters are separated by the bar symbol, |
- Each verb in the examples may be preceded by a label beginning in column 1.

ABEND

The ABEND statement may be used to abnormally terminate the execution of a CA Enterprise Tester Script. This command has the following format:

```
ABEND ( code )
```

Code (optional)

Can be an integer or a variable field that contains an integer from zero through 4095, to be used as an abnormal completion code.

ABEND

Abnormally terminates the execution of a script, using a specified or default abend code. If an abend code is not specified, a default of 4095 is used.

When a script is executed as a batch job step, execution of an ABEND statement causes abnormal termination of the batch job step, and may be used in conjunction with JCL parameters to control the execution of subsequent job steps.

When a script is executed interactively, the ABEND statement causes the script to stop and wait for instructions from the Interactive System terminal user.

Example:

```
ABEND 1
```

ALLOCATE

The ALLOCATE statement allocates a data set to a specified data definition, (DD), name within the current address space. The data set must be cataloged and can be a fixed or variable length sequential data set, a VSAM Key Sequenced Data Set, (KSDS), a member of a partitioned data set, (PDS), or a member of a generation data group, (GDG).

Scripts can process other types of data organizations such as a data base by using the CALL statement to access subroutines written in languages which support standard z/OS call conventions. This command has the following format:

```
ALLOCATE ddname DATASET dsnamedisposition
      OLD | SHR | PASS
      ( MEMBER memname )
      ( ERROR routine | * )
```

ddname

A DD name to which the specified data set is to be allocated. ddname may be the actual DD name, may be expressed as a literal or a variable field that contains the DD name, or may be a null-valued variable field to request that a unique DD name be assigned by the system, and returned in the specified variable field. A specified DD name may not begin with the characters, PTE.

DATASET

A required keyword and indicates that the following specification is the name of the data set to be allocated.

Dsname

The name of the data set to be allocated and may be specified as a literal value or a variable field that contains the data set name; the data set name may be the fully qualified name of a fixed or variable length sequential data set, a VSAM KSDS, a member of a PDS, or a member of a GDG where the relative generation number, zero or -n, is enclosed in parentheses.

disposition

Required and must be one of the keywords, OLDor SHR, indicating if the data set is to be allocated with exclusive or shared control. An optional keyword, PASS may be specified to bypass automatic unallocation when script execution ends.

MEMBER

A keyword that indicates that a member of a PDS is to be allocated and that the following specification is the name of the member. MEMBER is required when dsname is a PDS.

memname

The name of the member of a PDS to be allocated and may be specified as a literal value or a variable field that contains the member name.

Examples:

```
ALLOCATE SYSUT1 DATASET 'my.file.name' SHR
    ALLOCATE &DD DATASET &DSN  MEMBER &MEM OLD
    set &indd to '' /* initialize to nulls */  allocate &indd /* let system
    assign DD name */   dataset 'my.trxn.file' shr
```

CA Enterprise Tester provides automatic unusual condition processing when an optional condition keyword is specified. Each condition keyword is followed by the name of a script routine that is to receive control if the condition is encountered.

A condition handling routine may be expressed as a statement label or number, as a variable field that contains a statement label or number, or may be the special symbol, asterisk, *, to specify that execution is to be resumed at the next script statement.

The only condition keyword supported by the ALLOCATE statement is ERROR.

Note: ERROR routine specifies the script routine that is to receive control when an error is encountered during the execution of an ALLOCATE statement.

By default, a data set allocated using the ALLOCATE statement, not also unallocated by the script using a FREE statement, is automatically unallocated when script execution ends. When the PASS keyword is specified, automatic unallocation is bypassed, allowing a data set to remain allocated to a TSO address space.

When a null-valued variable field is used for the DD name to be allocated, a unique name will be assigned by the system, returned in the variable field and used for the allocation. Subsequent OPEN, READ, and other file processing statements may use the variable to refer to the allocated data set. This technique will eliminate conflicts caused when multiple scripts attempt to use the same DD name, for example:

```
set  &indd  to ''       /* variable set to nulls */
allocate   &indd . . .  /* system assigns name   */
open input &indd . . .  /* &indd now contains    */
read       &indd . . .  /* unique name           */
```

When ALLOCATE processing is successful, the variable field, PTERESP is set to the value 0000, indicating that the data set was successfully allocated. Following successful allocation of a data set, an OPEN statement must be executed prior to using other data set access statements to process the data set.

When a valid ALLOCATE statement is processed and data set allocation fails, the variable field, PTERESP, is set to the dynamic unallocation error reason code that indicates the reason for the allocation failure. Error reason code documentation may be found in Appendix A of the tutorial topics that are available when the HELP command is entered at the ISPF/PDF PRIMARY OPTION MENU. Following an allocation failure, if the ALLOCATE statement specifies an ERROR handling routine, an information message is logged and script execution is continued at the specified error handling routine.

When dynamic allocation is not attempted because of an invalid ALLOCATE request, or if allocation fails and no ERROR handling routine was specified, the error is handled according to the active HANDLE ILLOGIC specification.

Refer to PTERESP in PSL Variables for a description of data set access response codes and specific error codes, and to Data Set Access for an overview level description of CA Enterprise Tester data set access capabilities.

AND

The AND keyword is used to join conditions in IF and DO statements, and specifies that the conditions are true when both conditions are true. Refer to DO for details.

CALL

The CALL statement enables a CA Enterprise Tester Script to execute programs written in other languages that support standard z/OS call conventions, including COBOL and Assembler. The CALL statement may pass a fixed or variable length list of parameters to the called program. This statement has the following format:

```
CALL pgmname( USING parmlist )( VL )( REUSE )
```

pgmname

The name of the program to be called and may be expressed as a literal or as a variable field that contains the name of the program to be called.

USING

An optional keyword that indicates one or more subroutine parameters follow.

parmlist

A list of one or more parameters to be passed to the called program and may include any combination or number of literals and/or CA Enterprise Tester variable fields.

VL

An optional keyword specified when a program is called that processes a variable length parameter list, and causes the last parameter in the list to be identified to the called program.

REUSE

An optional keyword which specifies that a program is to be executed more than one time, and that the same copy of the called program is to be reused each time that the program is called.

Example:

```
CALL 'DATEPGM' USING 'REQCODE' &INDATA &RESPDATA
```

The CALL statement executes a named program. A CALL statement may pass parameters to the called program by specifying the USING keyword and a list of parameters that may include one or more literals or variable fields. Literal fields and CA Enterprise Tester defined read-only variable fields (variables whose names begin with PTE) may be specified to pass information to a called program, however, these parameters can not be modified by a called program.

User-defined variable fields (variables whose names begin with other than PTE), may be used to pass information to and to receive information from a called program. A called program may directly modify user-defined variable fields that are specified as CALL parameters. A script may process any returned information by examining the variable parameter field. Variable fields that are used as parameters that will be modified by a subroutine must be large enough to hold all of the information that will be placed in the variable field. This may be accomplished by setting the variable field to a large value prior to calling the subroutine, for example:

```
set &parm-field(,1,256) to all ''call 'program' using &parm-field
```

If a CALL statement does not specify the REUSE keyword, the program is invoked using an z/OS LINK macro statement. This means that the program is loaded, used and deleted each time that the program is called. When a program is called and the REUSE keyword is specified the program is loaded the first time the program is called. When the called program returns, it is retained in memory for subsequent calls. In addition to the improved performance that results from loading the program only one time, REUSE also allows a reusable program to keep working storage intact across program calls. As a result, subroutines that open data sets or sign on to data base systems may be called multiple times to perform a series of related requests. When a program that has been called using the REUSE parameter is again called without specifying the REUSE parameter, the program is deleted from memory. Therefore, every call to a program that requires that the program remain loaded for subsequent calls must specify the REUSE keyword.

Following the execution of a called program, the CALL statement places the return code from a called program in the CA Enterprise Tester variable field, PTERC.

CLOSE

The CLOSE statement ends processing of a user data set.

Format

```
CLOSE ddname
      ( ERROR routine | * )
```

ddname

A data definition (DD) name that identifies the data set to be closed; ddname may be the actual DD name or may be expressed as a literal or a variable field that contains the DD name; the DD name specified can not begin with the characters PTE. ddname must have been previously opened by executing an OPEN statement.

Example

```
CLOSE USERDD
```

CA Enterprise Tester provides automatic unusual condition processing when an optional condition keyword is specified. Each condition keyword is followed by the name of a script routine that is to receive control if the condition is encountered. A condition handling routine may be expressed as a statement label or number, as a variable field that contains a statement label or number, or may be the special symbol, asterisk, *, to specify that execution is to be resumed at the next script statement. The only condition keyword supported by the CLOSE statement is ERROR.

Note: ERROR routine specifies the script routine that is to receive control when an error is encountered during the execution of a CLOSE statement. If ERROR is not specified on a CLOSE statement, an ERROR routine specified on the associated OPEN statement will be used.

After a data set has been closed, other data set access statements may not be executed until another OPEN statement is executed.

During the execution of the CLOSE statement, the variable field, PTERESP is set to indicate the success of the request. An error handling routine may examine PTERESP to determine the exact cause of an error.

Refer to PTERESP in PSL Variables for a description of data set access response codes and specific error codes, and to Data Set Access for an overview level description of CA Enterprise Tester data set access capabilities.

COMPUTE

The COMPUTE statement may be used to evaluate complex algebraic expressions. Addition, subtraction, multiplication, division (including integer quotient and remainder functions), unary operations, and absolute value functions are supported. By default, intermediate and final results are rounded to 30 significant digits, if needed, and low order, zero fraction digits are discarded in a manner similar to a hand-held, floating-point calculator. Intermediate and final results may be calculated to a specified number of decimal places using one of three kinds of rounding.

By default, operations in an algebraic expression are evaluated in the standard order established by other languages, including COBOL and REXX. Parentheses may be used to bracket inner expressions to control the order in which arithmetic operations are performed.

Format

```
COMPUTE target = expression
( edit adverbs
( SIGN )
( PUNCTUATE )
( JUSTIFY LEFT | CENTER | RIGHT ) )
```

target

The target variable field or substring of a variable field in which the computed decimal result is to be stored. If the result is stored in a substring of the target variable, or if any of the keywords, SIGN, PUNCTUATE, or JUSTIFY, are used to edit the result, then target becomes or remains a character variable field. In any other case target becomes a numeric variable field.

=

A required symbol and indicates that expression follows.

expression

An arithmetic or algebraic equation to be evaluated, and consists of any logical string of arithmetic operations, data items, and optional parentheses to control the order of evaluation, as described in What a COMPUTE Expression Contains.

edit adverbs

Optional keywords that may follow expression and can be used to edit a computed result prior to storing the result in the specified target variable field or variable field substring.

- SIGN—Places an explicit leading sign, + or -, in the edited result.

- PUNCTUATE—Inserts commas (or periods if DECIMAL=COMMA was specified as an installation or run-time option), into the integer part of the result to separate groups of 3 integer digits beginning at, and to the left of the one's position.
- JUSTIFY–Positions the result in a specified substring of the target variable field. When JUSTIFY is specified, the target variable field expression must specify a target substring position and length.

For example:

```
compute &target(,1,10) = &a * &b justify left
```

places the result in a 10 character field that begins in position 1 of &target.

JUSTIFY LEFT causes the result to be aligned in the left-most position of the target field.

JUSTIFY RIGHT causes the result to be aligned in the right-most position of the target field.

JUSTIFY CENTER left-justifies the sign and right-justifies the number in the target field.

The editing capabilities of the SET statement and COMPUTE statement may be used to create formatted columner output. For an example of how to create columner output and more information on editing numeric data, refer to Edit Decimal Data.

Examples:

```
COMPUTE &sum           = 1,146,793 + 3,142 + 1,132.73
COMPUTE &product       = .065 * 12,345.67
COMPUTE &x             = (&A + &B) * (&C -&D)
COMPUTE &y(&i)         = (((&A + <&row &col 10>)
                       * (&B -&RECORD(,23,8))
                       * (&C / .5  ROUND DOWN 0) + &D) * 4)
COMPUTE &quo           = &A / QUOTIENT &B
COMPUTE &rem           = &A / REMAINDER &B
compute &total         = &total + <&row 60 10>
compute &edited(,1,10) = &total round 2 sign punctuate justify center
```

What a COMPUTE Expression Contains

An expression specified in a COMPUTE statement is an equation that consists of one or more data items and arithmetic operations that are evaluated to determine the decimal result that is subsequently stored in the specified target variable field.

You can use parentheses, to identify inner expressions that are to be evaluated prior to performing operations that precede or follow the inner expression. Additional pairs of parentheses may be used to identify inner expressions within higher-level inner expressions, to any number of levels. When an expression is evaluated, the lowest-level inner expressions are evaluated first and the intermediate result of each inner expression is used in place of the inner expression when evaluating higher-level expressions.

The value of all data items specified in a COMPUTE expression must be decimal values. Data items may be actual decimal values, or may be literals, variable fields, entries in variable field arrays, substrings of variable fields, or screen fields, that contain decimal values. Precisely, decimal values consist of the following characters:

- an optional leading sign, +or -.
- an optional 1 through 30 digit integer, which may be punctuated with commas (or periods if DECIMAL=COMMA is specified as an installation or run-time option), to separate groups of three integer digits beginning at and to the left of the ones position.
- an optional period (or comma if DECIMAL=COMMA is specified as an installation or run-time option), that represents a decimal point; if a decimal point is included in the value, at least one decimal digit must precede or follow the decimal point.
- an optional decimal fraction that follows a decimal point.
- a decimal value may be anywhere within a blank field and blanks may exist between a sign and the rest of the number; a field that contains only blanks or a variable field whose length is zero is considered to be equal to zero.

The Script Language SET statement may be used to convert other numeric data from binary, packed or zoned data formats to decimal format for use in a COMPUTE statement, and may be used to convert computed decimal results to binary, zoned or packed data formats.

The following arithmetic operations are supported by the COMPUTE statement:

- Arithmetic operators, + (add), - (subtract), * (multiply), / (divide), / QUOTIENT (divide and discard remainder), and / REMAINDER (divide and return remainder only), operate on the values of the data items or on the intermediate results of the inner expressions that precede and follow the arithmetic operator.
- ROUND may be used to specify the number of decimal fraction digits, from 0 to 30, that are to be retained in the intermediate result that precedes the ROUND keyword.
  - ROUND n, rounds the preceding intermediate result to n decimal places, adding 1 in the last retained digit if the first discarded digit is 5 or greater.

  - ROUND UP n, rounds the preceding intermediate result to n decimal places, adding 1 in the last retained digit if any discarded digit is greater than zero.
  - ROUND DOWN n, rounds the preceding intermediate result to n decimal places, discarding all decimal places beyond the retained digits, regardless of the value of the discarded digits.

  If the intermediate result being rounded has fewer than the specified number of decimal places, the intermediate result is rounded to n decimal places by adding a sufficient number of zero digits.

- Unary operators, - , and ABSOLUTE operate on the value of the data item, or on the intermediate result of an inner expression that follows. -changes the sign of the value or intermediate result, from positive to negative, or from negative to positive. ABSOLUTE makes the sign of the value or intermediate result, positive.

Because the symbol, - , may be a symbol for subtraction, a unary minus, or a hyphen in a variable field name, a blank should precede and follow - when the symbol is used as an arithmetic or unary operator in a COMPUTE expression, to avoid ambiguities like &N-1, which would be interpreted as a variable field name, and not as &N - 1. (However, &N -1or 1-&N are not ambiguous and are processed as &N - 1and 1 - &N).

Default Order of Multiple Operations within an Inner Expression

When parentheses are not used to specify the order in which operations in an expression are to be performed or when multiple operations are specified within an inner expression, then operations are performed in the following default order:

- Unary operations are performed left to right, except that consecutive unary operations are performed right to left so that:

  ```
  -ABSOLUTE  1 = - ABSOLUTE -1 = -1
  ```

- Multiplication and division are performed left to right.
- Addition and subtraction are performed left to right.
- ROUND implies an inner expression that includes all unparenthesized operations and all lower-level inner expressions that precede the ROUND keyword. For example:

  ```
  ( &a + &b ) * +   ROUND 0   + &d
  ```

  is evaluated as:

  ```
  ( ( ( &a + &b ) * + ) ROUND 0 ) + &d
  ```

DATE

The DATE statement provides the current date to a script. This statement has the following format

```
DATE
```

Execution of a DATE statement sets character variable fields to the current date elements:

- PTEMM will be set to a value from 01 through 12.
- PTEDD will be set to a value from 01 through 31.
- PTEYY will be set to the two-digit year.

DATE variable fields may not be set by a script, but may otherwise be used wherever other character variable fields may be used. The DATE variable fields are automatically set at the beginning of script execution, therefore it is necessary to issue the statement only when a change in date during the execution of a script is of significance.

Formatting and Calculating Dates

Any script to interpret, calculate may link to a distributed CA Enterprise Tester script, PTEDATE, and format dates from Jan 1, 0001 through Dec 31, 9999, in 101 different Gregorian, Julian and relative day date formats. PTEDATE handles numeric dates, dates with English, German, French or Dutch month and weekday names or abbreviations, and dates containing any possible punctuation.

For example, if the following script statements were executed on 3-1-07, the variable, &today, would be set to 'March 1, 2007'.

```
SET &DATE TO 'TODAY FORMAT &today ENGLISH "MONTH D, YYYY" '
LINK PTEDATE
```

If the following script statements were executed immediately after the preceding statements, 365 days would be added to March 1, 1995, the variable, &nextyear, would be set to the formatted result, 29 February 1996, and the variable, &weekday, would be set to Donnerstag, German for 29 February 1996 and Thursday.

```
SET &days TO +365
SET &DATE TO ' &today ENGLISH "MONTH D, YYYY" '
      ' PLUS &days '
      ' FORMAT &nextyear GERMAN  "D MONTH YYYY" '
           ' &weekday   "WEEKDAY" '
LINK PTEDATE
```

To Use PTEDATE:

1. Set the variable, &DATE, to a value, as explained below, that specifies and describes an input date, optionally specifies a number of days to be added to the input date, and specifies one or more output variables and output data formats.
2. If the 2-digit year mnemonic, YY, is used to define input dates that may be in a year other 1900 through 1999, then set the variable, &DATE-CCYY, to the lowest possible 4-digit year of the input date to be processed. For example, if an input date defined using YY is in a year from 1965 through 2064, then:

   ```
   SET &DATE-CCYY TO '1965'
   ```

3. Link to PTEDATE.
4. If variable, &DATE-ERROR, is greater than a blank, the value is a message that describes an invalid &DATE value, input date value or PLUS days value.

To Specify PTEDATE Request Parameters

```
SET &DATE TO ' TODAY
 &date1 lang1 "pic1"
  PLUS &days
FORMAT &date2 lang2 "pic2" &date3 lang3 "pic3"
 &daten langn "picn" '
```

- TODAY—Specifies that the input date is the current date. The first &DATE data must be TODAY or &date1.
- &date1—Any variable field that contains the input date described by lang1and pic1. The first &DATE data must be &date1 or TODAY.
- lang1—Optional, and specifies the language of any month name or abbreviation contained in &date1. lang1 can be English, German, French, Dutch or any other language coded in the PTEDATE language table, and defaults to the first language coded in the language table.
- pic1—A picture of the date contained in &date1 and is enclosed in single or double quotes. A picture is one or more input mnemonics listed in Table 6-1, that describe components of the date contained in &date1, and any blanks or special symbols less than C'a' contained in the date. Mnemonics, blanks and special symbols are coded in the order of the data in &date1. A single blank may be coded in pic1 to represent any number of or a variable number of blanks in &date1.
- A Gregorian date picture includes 3 date component mnemonics for year, month and day-of-month.
- A Julian date picture includes a year mnemonic and the ddd mnemonic for day-of-year.
- A relative day is the number of a day relative to day 1, January 1, 0001. A relative day picture includes just the day mnemonic.

PLUS

An optional keyword that identifies a variable, &days, that follows the keyword.

&days

Any variable that follows PLUS and contains a positive or negative decimal number of days to be added to the input date before formatting output date(s).

FORMAT

A required keyword that specifies that date data extracted from either TODAY or &date1, plus any specified PLUS &days, is to be formatted as specified by one or more format requests that follow the FORMAT keyword. Each format request includes a variable field name, an optional language and a picture that describes the data to be formatted and stored in the variable field that precedes the picture. Any number of format requests may be coded following FORMAT to format a date multiple ways or to store weekday, relative day or other date components in separate variable fields.

&date2 or any &date3 through &daten

The name of a variable field in which date data described by the language and picture that follow the variable field is to be formatted and stored.

lang2 or any lang3 through langn

Optional, and specifies the language of any month or weekday names or abbreviations to be stored in the previous output variable. Any lang2 through langn may be English, German, French, Dutch or any other language coded in the PTEDATE language table. Language defaults to the previous language specified, including lang1, or to the first language coded in the PTEDATE language table if no previous language has been specified.

pic2 or any pic3 through picn

A picture of the data to be stored in the output variable field that precedes the picture, and is enclosed in single or double quotes. Each picture is a string of one or more output date component mnemonics listed in Table 6-1, and any blanks or special symbols less than C'a' that are to be included in the output variable. The data stored in the output variable is formatted by concatenating components of the date referred to by mnemonics, with any blanks and special symbols, in the order in which they are coded in the picture. Any combination of mnemonics may be included in an output picture, however an output variable may be later processed as a &date1 only if it can be described by a pic1.

Date Mnemonic	Date Component	Date Data
yy	Year- number	00-99 (2000-2099)

Date Mnemonic	Date Component	Date Data
yyyy	Year- number	00001-9999
m	Month- number	1-12
zm	Month- number	' 1'-'12'
mm	Month-a number	01-12
mon	Month- name abbreviation	Jan-Dec in specified language
month/mn	Month- name	January-December in specified language
w	Weekday- number	1=Monday 7=Sunday
we	Weekday- name abbreviation	Mo-Su in specified language
wee	Weekday- name abbreviation	Mon-Sun in specified language
weekday/wn	Weekday- name	Monday-Sunday in specified language
d	Gregorian- day	1-31
zd	Gregorian- day	' 1'-'31'
dd	Gregorian- day	01-31
ddd	Julian- day	001-366
day	Relative- day	1=Jan 1, 001 3652059=Dec 31, 9999

DELAY

The DELAY statement causes a script to wait a specified amount of time between transactions, allowing outbound session traffic to flow during the delay, and may be automatically captured in a script created by the CA Enterprise Tester Record Facility. This statement has the following format:

```
DELAY hhmmssth |
   ( PERCENT factor )|
    INPUT | BRACKET
```

hhmmssth

A delay time interval, expressed in hours (hh), minutes (mm), seconds (ss) and tenths and hundredths of seconds (th). A delay time interval may be specified using a character variable field that contains the interval in the format, hhmmssth.

PERCENT

An optional keyword which specifies that all subsequent delay times should be factored by the specified rate.

factor

Can be an integer value from zero to 999, or may be a variable field that contains a number from zero to 999, and is the rate by which subsequent delay times will be multiplied.

INPUT

An optional keyword which specifies that delays specified by subsequently executed DELAY statements are to occur even when the session is in the in-bracket and inbound state.

BRACKET

An optional keyword which specifies that delays specified by subsequently executed DELAY statements are to occur only when the session is in the contention state and it is possible for the connected online system to start an SNA bracket.

Example:

```
/*           hhmmssth                                    */
DELAY            3200      /* 32 seconds                 */
DELAY              30      /* 3 tenths of a second       */
delay percent      50      /* pace script to enter transactions */
                           /* faster, only half of the delays   */
delay percent     200      /* pace script to enter transactions */
                           /* slower, twice the delay time      */
```

A recorded DELAY time represents the actual amount of time in an interactive session between the last message received from the online system and a subsequent request from the terminal user when an AID key is entered.

A DELAY statement may specify an amount of time, often called think time, that script execution should be delayed prior to sending a transaction entered by a subsequent Script Language TYPE statement. A CA Enterprise Tester script which contains actual recorded think time delays reproduces the session delays that occur on real 3270 sessions when a person gets a response from one transaction, then thinks about it or performs some other activity before entering the next transaction.

CA Enterprise Tester provides other Script Language statements which can control session event timing, including the THINK statement, which is documented in THINK , and the WAIT statement, documented in WAIT.

You may also refer to Multiple Session Testing for more information on how to control a work load presented to a system.

DELETE

The DELETE statement is used to delete one or more records from a VSAM Key Sequenced Data Set, (KSDS), identified by a full or generic key. This statement has the following format:

```
DELETE ddname KEY value
       ( NOTFOUND routine )
       ( ERROR routine | * )
```

ddname

The name of a data definition, (DD), statement that identifies the data set from which record(s) are to be deleted; ddname may be the actual DD name or may be expressed as a variable field that contains the DD name; ddname must have been previously opened by executing an OPEN statement; the DD name specified may not begin with the characters, PTE.

KEY

A required keyword and indicates that a key value follows.

value

A literal or variable field that contains a full or generic key value that identifies the record(s) to be deleted.

Example:

```
DELETE VSAMKSDS KEY &key  NOTFOUND nodelete
```

CA Enterprise Tester provides automatic unusual condition processing when an optional condition keyword is specified. Each condition keyword is followed by the name of a script routine that is to receive control if the condition is encountered. A condition handling routine may be expressed as a statement label or number, as a variable field that contains a statement label or number, or may be the special symbol, asterisk, *, to specify that execution is to be resumed at the next script statement. Condition keywords supported by the DELETE statement are NOTFOUND and ERROR.

- NOTFOUND routine specifies the script routine that is to receive control if a record is not found in the VSAM KSDS that has a key that matches the key value specified by the DELETE statement.
- ERROR routine specifies the script routine that is to receive control when an error is encountered during the execution of a DELETE statement. If ERROR is not specified on a DELETE statement, an ERROR routine specified on the associated OPEN statement will be used.

A DELETE statement that specifies a full key value deletes the VSAM record identified by the key. A DELETE statement that specifies a generic key value deletes every record in the VSAM KSDS whose key begins with the generic key value specified.

The DELETE statement sets the variable field, PTEDEL to the number of records deleted.

A NOTFOUND condition occurs if the VSAM KSDS contains no record with a key that matches the key value specified. If NOTFOUND handling is not specified, a NOTFOUND condition is considered to be an error and may be handled by an ERROR routine specified on the DELETE or OPEN statements.

During the execution of a DELETE statement, the variable field, PTERESP, is set to indicate the success of the request. An error handling routine may examine PTERESP to determine the exact cause of an error.

Refer to PTERESP in PSL Variables for a description of data set access response codes and specific error codes, and to Data Set Access for an overview level description of CA Enterprise Tester data set access capabilities.

DO

The DO statement identifies the beginning of an inline routine that may include any combination of Script Language statements. A DO routine is delimited by an ENDDO statement. A DO statement may specify one or more keywords and conditions to control the number of times the routine is repeated, and create logic structures within the script which support standard structured coding techniques. This statement has the following format:

```
DO ( ( BEFORE | AFTER ) WHILE |
     ( BEFORE | AFTER ) UNTIL |
       FOREVER
       first-expression
       EQ | NE | GT | GE | LE | LT |
       = |  > |  < |
      ª= | ª> | ª< |
      \= | \> | \<
      CONTAINS ( PREFIX | WORD | SUFFIX ) |
           second-expression |
      ( IS | IS NOT ) DECIMAL | PACKED | ZONED
      ( AND | OR other-expressions )
          routine-to-execute
      ( ITERATE | LEAVE )
ENDDO
```

BEFORE or AFTER

Optional keywords which specify that the DO procedure is to be performed before or after checking the specified WHILE or UNTIL conditions. The default is AFTER.

WHILE or UNTIL

Keywords which determine the number of times the routine will be executed, depending on specified conditional expressions which follow.

FOREVER

Can be used to create a routine which may be exited only by logic within the procedure, such as LEAVE, GO TO, or END.

first-expression, second-expression and other-expressions

Optional expressions for evaluation based on either the relational mnemonics or the relational symbols specified.

EQ, NE, GT, GE, LE and LT

Relational mnemonics for evaluation of expressions.

=, >, <, ª=, ª>, ª<, \=, \> and \<

Relational symbols for evaluation of expressions.

CONTAINS

Can be used to search for the specified value on a virtual terminal screen or in a variable field; and may be qualified by the optional adjectives, PREFIX, WORD or SUFFIX.

IS and IS NOT

Can be used to validate numeric data; and may be qualified by the optional adjectives, DECIMAL, PACKED or ZONED.

AND and OR

Keywords which further define relationships between expressions for evaluation; the relationships may be parenthesized to control the order in which they are evaluated to determine the truth of a set of conditions.

routine-to-execute

Any set of unlabeled Script Language statements that are to be executed each time that specified conditions indicate that the DO procedure is to be continued.

LEAVE

Can be used to leave the DO routine.

ITERATE

Can be used to repeat the DO routine at specified UNTIL or WHILE condition checking.

ENDDO

Delimits the DO routine.

Examples:

```
SET &ROW TO 0
DO WHILE &ROW < &PTEROWS
   SET &ROW UP 1
   LOG '|' <&ROW 1 &PTECOLS> '|'
ENDDO
SET &RETRIES TO 0
DO UNTIL <1 1 &PTECOLS> CONTAINS 'ISPF/PDF PRIMARY OPTION MENU'
     SET &RETRIES UP BY 1
     IF &RETRIES > 20
        GO TO PANEL-NOT-DISPLAYED
     TYPE ENTER
ENDDO
```

If a DO statement does not specify one of the keywords, WHILE, UNTIL or FOREVER, then the DO routine is executed one time.

When WHILE, UNTIL or FOREVER is specified, then one or more conditions must be specified.

A condition consists of two data items that are connected by one of the relational mnemonics, EQ, NE, GT, GE, LE, LT; the keyword, CONTAINS; or either the keyword, IS or the keywords, IS NOT. One or a logical pair of the relational symbols, =,>, <, ª (not), or \ (not), may be used as an alternative to relational mnemonics. Each of the pair of related data items may be a CA Enterprise Tester screen or variable field expression, a literal, or a number.

When one of a pair of compared arguments is an actual decimal number or numeric variable, and the value of the other argument is a decimal number, then the arguments are compared algebraically, otherwise the arguments are treated as character strings and corresponding characters are compared from left to right based on the EBCDIC collating sequence of the compared characters. When character strings are compared, if the arguments are not the same length, then the shorter of the arguments is considered to be padded with a sufficient number of blanks to make the fields be equal in length. Refer to Numbers for a precise description of decimal numbers, and to Variable Memory for a precise description of numeric and character variables.

When character strings are compared and are found to be unequal, then the variable field, PTENE is set to the number of the first position within the compared fields that is not equal, and the variable field, PTENEL is set to the net length of the longest of the fields being compared, beginning at the first unequal position. When character strings are found to be equal, PTENE and PTENEL are set to zero. When more than a single pair of character strings, connected by the keywords, OR or AND, are compared, then the values of PTENE and PTENEL reflect the results of the last comparison made during evaluation of the specified conditions.

The CONTAINS keyword may be used to search for a specified value on a virtual terminal screen or in a variable field. The CONTAINS condition is true when the argument being searched for is found in the specified screen or variable field. The search conducted when the CONTAINS keyword is used is able to find a desired value even when the alphabetic case of the argument being sought differs from the case in the searched field. The search sees all screen attributes, binary zeros and data in non-display screen fields as blanks, allowing screen search arguments to be based on what is seen on a screen.

When the CONTAINS keyword is used, the data item that follows is the argument being sought. The data item may be preceded by one of the adjectives PREFIX, WORD or SUFFIX to further define the value being sought. When PREFIX is used, only an occurrence of the search argument that is not preceded by, but is followed by an alphabetic or numeric character will be found. When WORD is used, only an occurrence of the search argument that is neither preceded, nor followed by an alphabetic or numeric character will be found. When SUFFIX is used, only an occurrence of the search argument that is preceded by, but is not followed by an alphabetic or numeric character will be found.

The CONTAINS condition provides feedback in variable fields. When a screen field is searched, variable fields PTEARGR and PTEARGC are set to the screen row and column at which the argument is found and PTENETR and PTENETC are set to the screen row and column of any net data, contained in the searched screen field, that follows the argument found. When a variable field is searched, variable field PTEARGI is set to the position within the variable field at which the argument is found and PTENETI is set to the position in the variable field of any net data contained in the searched variable field that follows the argument found. If an argument is found in any source field and more data follows the argument in the searched field, then PTENETL is set to the length of the net field data. When an argument is not found, all applicable feedback variables are set to zero. When no net data follows an argument in a searched field, all applicable net data fields are set to zero.

For example:

```
set &retries to 0
do until <1 1 &ptesize> contains 'ikj56700a enter userid -'
 set &retries up 1
  if &retries > 10
  then
      log 'USERID prompt not displayed.  I give up.'
      abend 1
  endif
  invite 1000
enddo
      type <&ptenetr &ptenetc> &userid enter
```

When one of the fields being compared is a TRACKED screen field that was read from a CA Enterprise Tester trace data set using the TRACK statement, unequal data may be considered equal if it is described by an active MASK statement. Refer to MASK for details.

The IS or the IS NOT condition may be specified to validate numeric data. The condition must be qualified by one of the adjectives, DECIMAL, PACKED, or ZONED

When more than one condition is specified, adjacent conditions are connected by one of the keywords, AND or OR. Conditions connected by AND must both be true for the pair of conditions to be considered true. A pair of conditions connected by OR are considered true if either of the conditions is true. By default, when three or more conditions are specified, consecutive conditions connected by AND must all be true. OR may be used to specify an alternative set of conditions. Parenthesis, (and), may be used to override the default order of evaluating multiple conditions.

For example:

```
do while  &v1 = 'a' and &v2 = 'b'  or &v3 = 'c'
```

is equivalent to:

```
do while (&v1 = 'a' and &v2 = 'b') or &v3 = 'c'
```

The following statement overrides the default order of evaluation:

```
do while &v1 = 'a' and (&v2 = 'b' or &v3 = 'c')
```

A DO routine may consist of any set of unlabeled Script Language statements, including additional DO statements that may be nested to any desired number of levels.

When a DO WHILE statement is executed, the specified conditions are evaluated before the DO routine is executed, unless the AFTER keyword is specified. If the set of conditions is true, the routine is executed. When the ENDDO or ITERATE keyword is encountered, the WHILE conditions are again evaluated, and the DO routine is executed again if the WHILE conditions continue to be true. When the WHILE conditions are determined to be false, script execution is resumed at the script statement that follows the ENDDO delimiter.

When a DO UNTIL statement is executed, the specified conditions are evaluated before the DO routine is executed, unless the AFTER keyword is specified. If the set of conditions is false, the routine is executed. When the ENDDO or ITERATE keyword is encountered, the UNTIL conditions are again evaluated, and the DO routine is executed again if the UNTIL conditions continue to be false. When the UNTIL conditions are determined to be true, script execution is resumed at the script statement that follows the ENDDO delimiter.

The ITERATE keyword and the LEAVE keyword are designed to simplify and expand the capabilities of IF/THEN/ELSE/ENDIF routines imbedded within DO procedures. The keywords may be used to create select structures that avoid multiple levels of nested IF routines with many trailing ENDIF statements. When ITERATE is specified, the procedure will repeat beginning at the DO condition checking, bypassing any remaining logic within the DO procedures. The LEAVE keyword will immediately leave the procedure and branch to the next executable statement following the ENDDO statement, (or follow other appropriate structure return rules if the DO procedure was the last logic in a PERFORM or LINK statement controlled routine.)

For example:

```
do forever
  read myindd into &record error *
  if &pteresp ne '0000'
  then
     leave/* leave do procedure */
endif                        /* and go to next line */
if &record(,1,3) = 'A10'
 then
  perform a10-routine
  iterate              /* skip rest of procedure */
endif                  /* and go back to do      */
if &record(,1,3) = 'B20'  then
  perform b20-routine
  iterate              /* skip rest of procedure */
 endif                 /* and go back to do      */
 if &record(,1,3) = . . .
then
      :
      :
enddo
```

DROP

The DROP statement deletes one or more named variable fields or variable field arrays from the CA Enterprise Tester variable pool. This statement has the following format:

```
DROP var1 ( var2 . . . )
```

var1, var2, varn

The names of the variable fields that are to be deleted from the CA Enterprise Tester variable pool. Because the DROP statement deletes an entire variable field or variable field array, specified variable field expressions may not include subscripts, substring positions or substring lengths. Variables fields beginning with PTE may not be dropped.

Example:

```
drop &end-date
drop &index &location-table
```

When a CA Enterprise Tester script begins execution, variable fields do not exist until a value is assigned to a variable field via a script PARM or by executing some Script Language statement that assigns a value to a variable. After a value is assigned to a variable, the variable exists until it is dropped. The assignment of a zero-length null value to a variable does not delete the variable from the CA Enterprise Tester variable pool. This distinction is significant when the propagation feature of the SET statement is used to re-initialize an entire variable field array.

After entries in a variable field array have been assigned values, even zero-length null values, the array must be dropped prior to using a SET statement to propagate a specific value into all entries of the array. This is because the SET statement propagates a given value only through non-existent lower numbered array entries that are created by a specific SET statement.

In the following example, an entire variable field array is deleted from the CA Enterprise Tester variable pool, after which a SET statement is used to assign a specified initial value to all 100 entries in the new array that is created by the SET statement.

```
DROP &ARRAYSET &ARRAY(100) TO 0
```

Refer to Propagate Data in a Variable Array for a description of the Script Language SET statement syntax for propagating data in an array.

DUMP

The DUMP statement may be used to log the current CA Enterprise Tester virtual terminal screen, to display the value of all variables currently in the CA Enterprise Tester variable pool, or to display the number of transactions executed, response time, and think time statistics for a session that is currently active. The specified data is displayed via the CA Enterprise Tester Log. This statement has the following format:

```
DUMP SCREEN | POOL | CLOCKS
```

SCREEN

A keyword which will log the contents of the virtual terminal screen

POOL

The default and will log the contents of the CA Enterprise Tester variable pool.

CLOCKS

Logs the current session statistics.

The DUMP statement can be used for diagnostic purposes to log the contents of memory that is available to a CA Enterprise Tester script. If no options are specified or if the POOL keyword is specified, then the DUMP statement logs the CA Enterprise Tester variable pool. When the SCREEN keyword is specified, the DUMP statement logs an image of the current virtual terminal screen, provided that a session is active. When the CLOCKS keyword is specified, the DUMP statement logs current session statistics, provided that a session is active.

A SCREENDUMP includes the names of the session partners and the current cursor position. The screen buffer is edited by converting attributes, binary zeros and data contained in non-display screen fields to blanks.

A POOL DUMP logs the CA Enterprise Tester variable pool. For each variable field, the dump displays the variable name, occurrence, type, numeric or character, and the contents of the variable field. For character variables, variable field data is displayed in both character and hexadecimal formats.

When a DUMP statement that specifies CLOCKS is executed and a session is active on which more than one transaction has been executed, message PTE10497 is issued. Message PTE10497 is also issued automatically prior to normal or abnormal termination of a session. The message contains the number of transactions executed and a table of response time and think time statistics. Refer to message PTE10497 in CA Enterprise Tester Messages for a description of the statistics displayed.

ELSE

The ELSE keyword is used in an IF statement to identify a routine that is to be executed when specified conditions are false. Refer to IF for details.

ENCODE

The ENCODE statement allows a script to control security coding of traced or recorded modifiable dark data. The ENCODE statement may control whether or not modifiable dark data is to be encoded or may specify the user ID that is to be used in the encoding process. This statement has the following format:

```
ENCODE ON | OFF | FOR userid
```

ON

Specifies subsequent recorded modifiable dark data should be encoded.

OFF

Specifies subsequent recorded modifiable dark data should be captured without encoding.

FOR userid

Specifies subsequent recorded modifiable dark data should be encoded for the named user ID.

By default, when a FORMAT TRACE is produced or when a recording is created by the CA Enterprise Tester Interactive System, the first word of each modifiable dark field is encoded so that passwords are not revealed to unauthorized persons. The ENCODE statement allows a script to turn the encoding feature on or off or to specify the user ID for which data is to be encoded. When a user ID is specified, the user ID may be specified via a literal or variable field that follows the FOR keyword.

When password encoding is turned off, actual passwords entered on a traced or recorded session are captured in the trace or recording. A system option is provided that may be used to disable the ability to turn encoding off.

Refer to PSL Reference for complete information on CA Enterprise Tester security features.

END

The END statement causes execution of a CA Enterprise Tester Script to end normally. This statement has the following format:

```
END rc
```

rc

Optional and may be an integer or a variable field that contains an integer from zero through 4095, to be used as the job step return code.

Examples:

```
ENDEND 8END &RC
```

If an END statement return code is not specified, zero is assumed. When a script is executed via a batch job step, the return code option may be used to control the execution of subsequent job steps through the appropriate use of the JCL COND parameter.

If a session is active when an END statement is executed via a batch job step, then the session is automatically terminated prior to ending script execution. When a script is played via the Interactive System, any active session is left intact for use by the Interactive System user.

An END statement is automatically added at the end a played script or at the end of a script read from the PTEIN data set(s).

ENDDO

The ENDDO verb identifies the end of a DO statement routine. Refer to DO for details.

ENDIF

The ENDIF verb identifies the end of an IF statement ELSE routine or the end of an IF statement THEN routine, when an ELSE routine is not specified. Refer to IF for details.

ENDRECORD

The ENDRECORD verb ends an active session recording and closes the recorded script data set. This verb has the following format:

```
ENDRECORD
```

ENDTRACE

The ENDTRACE verb ends an active session trace and closes the trace data set. This verb has the following format:

```
ENDTRACE
```

EXECUTE

The EXECUTE statement specifies a list of one or more CA Enterprise Tester data items, including literals, variable fields, and other CA Enterprise Tester data items. When an EXECUTE statement is processed, the values of the specified data items are dynamically concatenated to construct another Script Language statement. The constructed statement is then executed immediately. This statement has the following format:

```
EXECUTE string
```

string

A free-form list of data items that represent the elements of the Script Language statement for concatenation and immediate execution.

Example

```
EXECUTE 'set &' &target-field ' to &source-field'
```

If a variable field, &TARGET-FIELD, contains the value OPTION, and a variable field, &SOURCE-FIELD, contains the value, ADD, then execution of the following statement:

```
EXECUTE 'SET &' &TARGET-FIELD ' TO &SOURCE-FIELD'
```

constructs a new Script Language statement by concatenating the values of the specified literal and variable data items to yield:

```
SET &OPTION TO &SOURCE-FIELD
```

The constructed statement is executed immediately. Following execution of the constructed statement, the value of the variable, &OPTION, would be ADD.

The EXECUTE statement may be used to perform indirect addressing, may be used to create variable fields whose names come from data processed by the script, or may be used to support other complex coding techniques.

The Script Language statement constructed by the EXECUTE statement must begin with a CA Enterprise Tester verb and may contain only one verb.

The EXECUTE statement may not be used to execute the following verbs:

- IF, AND, OR, THEN, NEXT, ELSE, ENDIF
- DO, UNTIL, WHILE, ENDDO
- LINK
- EXECUTE

EXIT

The EXIT verb causes script execution to be resumed at the statement that follows the most recently executed PERFORM or LINK statement. This verb has the following format:

```
EXIT
```

When a PERFORM or LINK statement is executed, the location of the statement that follows the PERFORMor LINK is remembered. Performed and linked-to subroutines may PERFORM or LINK to other routines, causing a list of return points to be remembered. When an EXIT verb is executed, the most recent return point is removed from the list and script execution is resumed at the remembered location.

An EXIT verb is automatically added as the last statement of every script that is invoked via a LINK statement.

FLOW

When a script is being executed, statements may be optionally logged just before they are executed. The FLOW statement allows a script to control script statement execution logging. This statement has the following format:

```
FLOW ON | OFF
```

ON

Specifies subsequent script statements should be logged before execution.

OFF

Specifies subsequent script statements should not be logged.

The FLOW statement is provided as a debugging aid. Script statement execution logging may be controlled externally by specifying FLOW or NOFLOW as a JCL EXEC statement PARM field or via the PARM field provided on the Interactive System Play Panel.

The FLOW statement may be used in a script to selectively turn execution time script statement logging on or off.

FREE

The FREE statement unallocates the data set associated with a specified DD name. This statement has the following format:

```
FREE ddname  ( ERROR routine | * )
```

ddname

A data definition, (DD), name associated with the data set that is to be unallocated; ddname may be an actual DD name or may be expressed as a literal or a variable field that contains the DD name; the DD name specified may not begin with the characters PTE.

Examples:

```
FREE SYSUT1FREE &DD
```

CA Enterprise Tester provides automatic unusual condition processing when an optional condition keyword is specified. Each condition keyword is followed by the name of a script routine that is to receive control if the condition is encountered.

A condition handling routine may be expressed as a statement label or number, as a variable field that contains a statement label or number, or may be the special symbol, asterisk, *, to specify that execution is to be resumed at the next script statement. The only condition keyword supported by the FREE statement is ERROR.

ERROR routine specifies the script routine that is to receive control if an attempt to free a data set fails.

After processing a data set, a CLOSE statement must be executed prior to executing a FREE statement to unallocate the data set.

When a FREE statement completes successfully, PTERESP is set to 0000, indicating that the data set is no longer allocated.

When a valid FREE statement is executed and unallocation fails, the variable field, PTERESP, is set to the dynamic unallocation error reason code that indicates the reason for the unallocation failure. Error reason code documentation may be found in Appendix A of the tutorial topics that are available when the HELP command is entered at the ISPF/PDF PRIMARY OPTION MENU. Following an unallocation failure, if the FREE statement specifies an ERROR handling routine, an information message is logged and script execution is continued at the specified error handling routine.

When dynamic unallocation is not attempted because of an invalid FREE request, or if unallocation fails and no ERROR handling routine was specified, the error is handled according to the active HANDLE ILLOGIC specification.

Refer to PTERESP in CA Enterprise Tester Defined Variables for a description of data set access response codes and specific error codes, and to Data Set Access for an overview level description of CA Enterprise Tester data set access capabilities.

GOTO

The GOTO statement, which may optionally be spelled GO TO, causes script execution to be continued at a specified script statement. This statement has the following format:

```
GOTO statement
```

statement

Identifies the next script statement to be executed and may be expressed as an actual script label or statement number or may be a variable field that contains a script label or statement number.

Example:

```
GOTO ABENDIT
```

HANDLE

The HANDLE statement can be used to control how particular session events are handled, may be used to activate session event handling options that give a script control of messages and other session events not normally handled by a script, or may be used to control the handling of errors that occur during script execution. This statement has the following format:

```
HANDLE event ON | OFF |
      AT statement | label
      event
          CONTENTION
          EVENTS
          ILLOGIC
          LOGONFAIL
          OUTBOUND
          RECONNECT
          RESPONSES
          SESSION
          FAIL
          TIMEOUT
          TURNAROUND
```

EVENTS, EVENTS, ILLOGIC, LOGONFAIL, OUTBOUND, RECONNECT, RESPONSES, SESSIONFAIL, TIMEOUT or TURNAROUND

Specifies the type of session event that is to be handled;

ON, OFF or AT

Specifies how the condition is to be handled. If an option is not specified, then ON is assumed. When AT statement is specified, statement may be an actual script statement number or may be a variable field that contains a script statement number; when AT label is specified, label may be an actual script label or may be a variable field that contains a script label.

Example:

```
HANDLE LOGONFAIL AT LOGONERR
```

CONTENTION

A HANDLE CONTENTION statement provides one way for a script to control the flow of transactions when both script and the online system attempt to send a message at the same time. The CONTENTION condition can arise only while PTESTATE is ONLINE or CONTENTION.

HANDLE CONTENTION ON

Specifies that when a TYPE statement is executed, if the online system has attempted to initiate a bracket, then permission should be granted. After receiving the message sent by the online system and after placing the message into the CA Enterprise Tester screen buffer, the TYPE statement is executed as though no contention had occurred.

HANDLE CONTENTION OFF, (the default)

Specifies that any attempt by the online system to initiate a bracket is to be rejected by CA Enterprise Tester, unless the script grants specific permission to the online system to initiate a bracket by executing an INVITE statement.

HANDLE CONTENTION AT

Statement specifies that when a TYPE statement is executed, if the online system has requested permission to initiate a bracket, then permission should be granted. After the online system message has been received, TYPE statement processing is ended prior to keying the data specified on the TYPE statement. When a named script routine is entered after processing a contention condition, the variable field, PTELINE, is set to the script line number that was being executed when the message was received, and PTENEXT is set to the script line number that would have been executed next if the HANDLE statement had not caused a branch.

EVENTS

A HANDLE EVENTS statement may be used to process random session events asynchronously. That is, HANDLE EVENTS provides a way for a script to handle random messages from an online system when they occur, regardless of where the script is executing.

HANDLE EVENTS ON

Specifies that when a message from an online system is detected, the message is to be received and processed immediately and automatically. Detection may occur between the execution of any two script statements. A detected session event is processed as though an INVITE statement had been used to process the event. The virtual terminal screen is updated if a message is received and all appropriate session related variables are updated. Particular events are processed considering event handling specified by other HANDLE statements. If another HANDLE statement does not specify an action to be taken when a particular event is processed, then script execution is resumed at the statement that follows the statement that was executing when the event was detected.

HANDLE EVENTS OFF, (the default)

Specifies that online system messages and other session events are to be processed only when TYPE or INVITE statements are being processed.

HANDLE EVENTS AT

Statement works exactly like HANDLE EVENTS ON, except that after processing a detected event, script execution is resumed at a specified script statement. When a named script routine is entered after processing an event, the variable field, PTELINE, is set to the script line number that was being executed when the event was detected, and PTENEXT is set to the script line number that would have been executed next had the event not occurred.

ILLOGIC

A HANDLE ILLOGIC statement specifies the action to be taken when an error occurs while processing a Script Language statement and the error is not handled by another error handling feature.

HANDLE ILLOGIC ON

Instructs CA Enterprise Tester to issue a message documenting the problem, but to ignore the error and resume script execution at the next statement.

HANDLE ILLOGIC OFF, (the default)

Specifies that script execution is to be stopped in the event that an illogical script statement is executed. If an ILLOGIC condition is encountered during the execution of a script invoked via a batch job step, then the job step is abnormally terminated. An ILLOGIC error encountered during the execution of a script invoked via the Interactive System PLAY command, causes script execution to be stopped. Subsequent action is the decision of the Interactive System terminal user.

HANDLE ILLOGIC AT

Statement identifies an error routine that is to be given control when an otherwise unhandled error is encountered. When a named script routine is entered after processing an error, the variable field, PTELINE is set to the script line number that was being executed when the error was detected, and PTENEXT is set to the script line number that would have been executed next had the error not occurred. When the error handling routine is entered, ILLOGIC handling is turned off, to avoid recursive entry to the error handling routine in the event that the routine also contains logic problems. If an ILLOGIC error handling routine should remain active, even after an ILLOGIC error is encountered, the error handling routine may begin with a HANDLE ILLOGIC statement that reactivates the error handling routine.

LOGONFAIL

A HANDLE LOGONFAIL statement specifies the action to be taken if execution of a LOGON statement does not complete normally.

HANDLE LOGONFAIL ON

Specifies that CA Enterprise Tester should automatically prompt the console operator for assistance if an attempted LOGON fails. The prompt consists of three messages.

- The first provides relevant information about the problem encountered.
- The second message documents the VTAM names of the CA Enterprise Tester virtual terminal and the online system with which a session could not be obtained.
- The third message solicits a reply of either RETRY, DOWN, or REJECT.

If the operator is able to correct the problem and replies RETRY, the LOGON attempt is automatically retried. If the retry is successful, the script LOGON statement will complete normally with no further script action. If the retry fails, the prompt is reissued. If the operator replies DOWN, meaning that the online system is down, or replies REJECT, because the cause of the problem can not be determined or corrected, the variable field, PTEREPLY is set to the operator's reply, and the condition is handled according to the active handling for SESSIONFAIL.

If HANDLE LOGONFAIL ON is executed within a script invoked under the control of the Interactive System PLAY command, then a logon failure will be treated as an error, no message is delivered to the console operator, and script execution is stopped. The Interactive System terminal user may determine the next action.

HANDLE LOGONFAIL OFF, (the default)

Specifies that an unsuccessful attempt to obtain a session should be treated as a session failure and be handled according to the active specification for the SESSIONFAIL condition.

HANDLE LOGONFAIL AT

Statement identifies a script routine that is to be given control when a LOGON fails. When a named script routine is entered following a LOGON failure, the variable field, PTELINE, is set to the script line number that was being executed when the error occurred, and PTENEXT is set to the script line number that would have been executed next had the error not occurred.

OUTBOUND

A HANDLE OUTBOUND statement may be used by a script to gain access to online system messages that do not free the terminal keyboard. An outbound message is a message sent by an online system that does not end an SNA bracket and does not change the session direction to inbound. An outbound message leaves a session in the outbound state.

HANDLE OUTBOUND ON, (the default)

Specifies that TYPE or INVITE statement processing is not to be completed by a message that leaves the session in the outbound state. Rather, when an outbound message is received, the message is placed in the virtual terminal buffer and the next online system message is automatically received. When a message is received that frees the terminal keyboard, a TYPE or INVITE statement completes. The variable field, PTECC indicates the number of messages that were received during TYPE or INVITE statement processing.

HANDLE OUTBOUND OFF

Inactivates default handling of outbound messages. After turning default outbound message handling off, the execution of a TYPE or INVITE statement will complete when an outbound message is received and placed in the virtual terminal buffer. When a TYPE or INVITE statement is completed by receiving an outbound message, the variable field, PTESTATE is set to the value OUTBOUND. Script execution is then resumed at the next statement. When PTESTATE is OUTBOUND, a script may receive another online system message by executing an INVITE statement. While in the outbound state, the variable field, PTEKB, contains the value INHIBITED, indicating the state of the virtual terminal keyboard. While the keyboard is inhibited, a TYPE statement may be used only to send an Attention key signal by specifying the TYPE statement keyword, ATTN.

HANDLE OUTBOUND AT

Statement works exactly like HANDLE OUTBOUND OFF, except that after receiving an outbound message, script execution is resumed at a specified script statement rather than at the statement that follows the TYPE or INVITE statement.

RECONNECT

A HANDLE RECONNECT statement specifies the action to be taken in the event that a previously obtained session is passed by the online system to another online system or in the event that a session has ended and a queued session with another online system has been accepted. Refer to Passed and Queued Sessions for a full explanation of these events.

HANDLE RECONNECT ON, (the default)

Specifies that passed and queued sessions are to be accepted. When a session is passed by an online system, or when a session ends and a queued session is accepted, PTEPLU is updated to contain the name of the new online system, PTESTATE is set to ONLINE, and script execution is resumed at the script statement following the TYPE or INVITE statement at which the condition was encountered. The CA Enterprise Tester virtual terminal screen is cleared as a result of the reconnection. A script can recognize the condition by testing the variable fields PTEPLU and/or PTESTATE.

HANDLE RECONNECT OFF

Specifies that passed and queued sessions are not acceptable. When a passed or queued session is accepted, then the new session is automatically terminated. After termination of the new session, the condition is further handled according to the handling specified for ILLOGIC.

HANDLE RECONNECT AT

Statement specifies that passed and queued sessions are to be accepted. When a passed or queued session is accepted, PTEPLU is updated to contain the name of the new online system, and PTESTATE is set to ONLINE. When a named script routine is entered after accepting a passed or queued session, the variable field, PTELINE is set to the script line number that was being executed when the session was accepted, and PTENEXT is set to the script line number that would have been executed next had the event not occurred.

RESPONSES

A HANDLE RESPONSES statement provides options that may be used to process an unpredictable number of responses by an online system to terminal input from a recorded script.

HANDLE RESPONSES ON

Provides a way to wait for and automatically receive a variable number of responses from an online system following the execution of LOGON and TYPE statements that leave the session in the contention state.

HANDLE RESPONSES ON changes the operation of the INVITE statement. When response handling is turned on, a remembered invite interval is set to zero. Automatic processing of contention state outbound messages begins when a subsequent INVITE statement sets a non-zero invite interval. When response handling is on, an INVITE statement may be executed before or after starting a session, to specify an automatic invite interval. If the INVITE statement that activates response handling is executed while an active session is in the contention state, then automatic message processing is performed immediately. Once activated, response handling is performed each time an active session enters the contention state during the execution of LOGON, TYPE, or INVITE statements. When the contention state is entered, any messages sent by an online system within the invite interval are automatically received before the LOGON, TYPE, or INVITE statement completes.

When response handling is on, and after a non-zero invite interval has been established, an INVITE statement that is executed while a session is in the contention or inbound state serves only to alter the active invite interval. As a result, recorded INVITE statements need not be deleted from a recorded script when HANDLE RESPONSES ON is used to improve the operation of the script.

When response handling is on, INVITE statements executed while a session is in other than the contention or inbound states operate as though response handling were off. However, after performing normal INVITE processing, if response handling is on and a non-zero invite interval is specified, then automatic outbound message processing is also performed if the session enters the contention state.

If an INVITE statement is executed while response handling is on, and a zero interval is specified, then automatic contention state outbound message processing is suspended before performing normal INVITE statement processing. Automatic contention state outbound message processing is resumed when a subsequent INVITE statement is executed that specifies a non-zero interval.

When certain IMS response modes are used, IMS does not always respond to the same terminal input in the same way. Depending on other work going on in the system, IMS may queue the terminal input and free the terminal keyboard, then send the desired response at a later time. At another time, IMS may prepare and send the desired response immediately, without freeing the keyboard.

By default LOGON, TYPE or INVITE statements end when the keyboard is freed. If extra free keyboards are received when a recorded script is executed, a TYPE statement may execute before the appropriate input panel is displayed. HANDLE RESPONSES ON may be added to the script, or to a script that links to a recorded script, to handle the extra free keyboards automatically, without changing the recorded script statements. Recorded INVITE statements need not be deleted. When they are executed while HANDLE RESPONSES is ON, the first INVITE activates automatic response handling, and others simply reset the invite timer.

Unpredictable responses may be sent by IMS or by other online systems. For example, the following script is a recorded Net view logon, signon, and logoff. As is, the script may not always work, because Net view queues each terminal input, frees the keyboard, then at a later time sends the desired response to each input in an unpredictable number of pieces, freeing the terminal keyboard after sending each part of the response.

```
LOGON NETV
      MODEL D4A32XX3 24X80 43X80 EXTDS
INVITE 1000                /* WAIT FOR MESSAGE (BRACKET)  */
INVITE 1000                /* WAIT FOR MESSAGE (BRACKET)  */
TYPE <15,29> "rs1"
     <16,29>  CODE "VIF#5" /* CODED FOR USERID RS1       */
     <16,34>  ENTER
INVITE 1000                /* WAIT FOR MESSAGE (BRACKET)  */
INVITE 1000                /* WAIT FOR MESSAGE (BRACKET)  */
INVITE 1000                /* WAIT FOR MESSAGE (BRACKET)  */
INVITE 1000                /* WAIT FOR MESSAGE (BRACKET)  */
INVITE 1000                /* WAIT FOR MESSAGE (BRACKET)  */
TYPE <43,1> "logoff"
     <43,7> ENTER
INVITE 1000                /* WAIT FOR MESSAGE (BRACKET)  */
INVITE 1000                /* WAIT FOR LOGOFF (UNBIND)    */
```

Assuming that the name of the recorded script shown above is NETV, the following script may be used to alter the handling of responses and operation of INVITE statements, to make the script work correctly, regardless of the number of messages sent in response to each terminal input.

```
HANDLE RESPONSES ONLINK "NETV"
```

HANDLE RESPONSES OFF, (the default)

Requires that recorded or user-written scripts anticipate and deal with online system responses that are sent by an online system after a previous response was sent that freed the terminal keyboard. Recorded scripts contain INVITE statements that are intended to receive these delayed responses when the script is executed. However, some online systems send an unpredictable number of responses to a terminal input, and as a result, recorded scripts do not work correctly when more responses to a terminal input are sent when a recorded script is executed than were sent when the recorded script was captured.

HANDLE RESPONSES AT

Can be used by a script to specify a routine that is to be given control following the normal completion of every subsequently executed LOGON, TYPE or INVITE statement. When the routine is entered, the CA Enterprise Tester defined variable field, PTENEXT is set to the line number of the statement that follows the LOGON, TYPE or INVITE statement just executed. The routine may remember PTENEXT in another variable, may alter RESPONSES handling, may examine the virtual terminal screen or other available information, and may use TYPE or INVITE statements to manipulate a session in any desired way. After doing whatever is desired, the routine may restore HANDLE RESPONSES AT label processing and GO TO the remembered PTENEXT.

HANDLE RESPONSES AT may be used to add intelligence to a recorded script, without modifying the recorded script. For example, while HANDLE RESPONSES ON can be used to receive unpredictable online system responses automatically, a penalty is that the script must wait for some period of time to give the online system a chance to send an unknown number of messages. Sometimes, the wait may be unnecessary.

Using HANDLE RESPONSES AT processing, a routine may be written that automates response handling, accomplishing the same thing as HANDLE RESPONSES ON, but that also is intelligent enough to skip the wait for a response if it can determine that the expected response has already been received. A routine designed to handle responses for recorded scripts, can centralize and automate unpredictable response handling, without paying the penalty of slowing the scripts down.

For example, assume that a transaction named UPDT, which might be queued before freeing the keyboard, or might be processed without freeing the keyboard, displays the message REQUEST PROCESSED on line 24, when transaction processing has been completed. If a recorded script, script2, accesses the transaction, the following script could be used to intelligently decide if the transaction was processed immediately and wait for the desired response only when necessary.

```
script1
-------
      handle responses at check
      link "script2"
      end
check set &next to &ptenext
      if tracked <1 2 *> eq "UPDT"
          and <24 1 *> ne "REQUEST PROCESSED"
      then
          handle responses off
          do until <24 1 *> = "REQUEST PROCESSED"
             invite 1000
          enddo
          handle responses at check
      endif
      go to &next
script2 (may be a recorded script)
-------
       .
       .
       .
       TYPE <1 2> "TRN1" ENTER
       TYPE <1 2> "UPDT" ENTER
       TYPE <1 2> "TRN1" ENTER
       TYPE <1 2> "UPDT" ENTER
       .
       .
       .
```

SESSIONFAIL

A HANDLE SESSIONFAIL statement specifies the action to be taken if a session is abnormally lost during script execution.

HANDLE SESSIONFAIL ON, (the default)

Specifies that script execution is to be continued when a session is abnormally lost. If a session is lost, a message is logged, the variable field, PTESTATE, is set to OFFLINE, and script execution is resumed at the next script statement.

HANDLE SESSIONFAIL OFF

Specifies that if a session failure is detected during the execution of a script that is invoked via a batch job step, then script execution is to be abnormally terminated. A session failure detected during the execution of a script invoked via the Interactive System PLAY command causes script execution to be stopped at error. The error is handled according to commands entered by the Interactive System terminal operator.

HANDLE SESSIONFAIL AT

Statement identifies a script routine that is to be given control when a session failure is encountered. When a named script routine is entered following a session failure, the variable field, PTELINE, is set to the script line number that was being executed when the session failed, and PTENEXT is set to the script line number that would have been executed next had the session not failed.

TIMEOUT

A HANDLE TIMEOUT statement is used to control session time out processing. By default, CA Enterprise Tester waits indefinitely for the completion of session events. Time out processing may be used to limit the amount of time that CA Enterprise Tester will wait for the completion of any session event. HANDLE TIMEOUT only activates or inactivates time out processing. A separate statement, TIMEOUT, is used to specify the maximum time that CA Enterprise Tester should wait for the completion of any in-progress session event, when time out processing is active.

HANDLE TIMEOUT ON

Specifies that CA Enterprise Tester should wait no longer than the time interval specified on a TIMEOUT statement for the completion of any in-progress session event. If time out processing is active when a LOGON statement is executed, a session will not be started if LOGON statement processing can not be completed within the time out interval. When CA Enterprise Tester is waiting on the receipt of a message or the completion of another session event during TYPE or INVITE statement processing, the statement will complete before the completion of the session event if the event does not complete within the active time out interval.

Time out processing applies only to session events that are in-progress or expected. An INVITE time interval allows time for an online system to prepare and send a message. When an INVITE statement is executed in the contention state, a message is possible. However a session event is not in-progress until the first part of a message or other session event has actually been received. The INVITE interval allows time for a contention state event to begin. Time out processing begins immediately if an INVITE statement is executed in a state other then contention, or when the beginning of a contention state event is detected.

When a session times out during TYPE or INVITE statement processing, HANDLE TIMEOUT ON specifies that script execution is to resume at the statement that follows the timed out statement. The session may be in any of several states not normally seen by a script. The value of the variable, PTESTATE, identifies the state of the session. CA Enterprise Tester Logical Unit States documents all of the possible session states and the possible values of PTESTATE.

While a session is timed out, an INVITE statement may be used to resume the wait for the completion of the timed out event. Another variable field, PTEKB, identifies the state of the virtual terminal keyboard. While a session is timed out, the keyboard will be in either a LOCKED or INHIBITED state. When the keyboard is LOCKED, the TYPE statement may not be used. If the keyboard is in the INHIBITED state, only a TYPE statement that specifies the ATTN keyword may be used to send an Attention key signal to the online system.

HANDLE TIMEOUT OFF, (the default)

Specifies that CA Enterprise Tester should wait indefinitely for the completion of all session events.

HANDLE TIMEOUT AT

Statement works exactly like HANDLE TIMEOUT ON, except that when a session times out, script execution is resumed at a specified script statement, rather than at the statement that follows the timed out statement.

TURNAROUND

A HANDLE TURNAROUND statement controls the operation of the Script Language TYPE statement. The TURNAROUND keyword refers to the point in a session that occurs when a TYPE statement has sent input to an online system and changes the session direction to outbound.

HANDLE TURNAROUND ON, (the default)

Specifies that after changing the session direction to outbound, the TYPE statement waits for and receives the online system response before completing.

HANDLE TURNAROUND OFF

Specifies that a TYPE statement will complete after changing the session direction to outbound, and does not wait for a response. When HANDLE TURNAROUND OFF is active, a TYPE statement becomes a send-only operation. The INVITE statement is subsequently used to receive online system responses.

HANDLE TURNAROUND AT

Statement is similar to HANDLE TURNAROUND OFF except that script execution is resumed at a specified script statement after each TYPE statement is executed, and the INVITE statement is required to receive online system responses.

IF

The IF statement can be used to conditionally execute one or more Script Language statements. The IF statement has two forms, a Standard Form and a Compatibility Form.

The Standard Form IF statement evaluates one or more conditions and conditionally executes one of two in-line routines, depending on whether the conditions are true or false. The two alternative routines are identified by the keywords, THEN and ELSE. A Standard Form IF statement must be delimited by an ENDIF keyword. This statement has the following standard format:

```
IF first-expression
   EQ | NE |
   GT | GE | LE |LT |
= |  > | < |
   ª= | ª> | ª< |
   \= | \> | \<
   CONTAINS (PREFIX | WORD | SUFFIX) |
   second-expression |
  (IS | IS NOT)
   DECIMAL | PACKED | ZONED
  (AND | OR other-expressions)
 THEN true-routine
   (NEXT )
(ELSE false-routine )
   (NEXT )
ENDIF
```

first-expression, second-expression and other-expressions

Optional expressions for evaluation based on either the relational mnemonics or the relational symbols specified;

EQ, NE, GT, GE, LE, and LT

Relational mnemonics for evaluation of expressions

=, >, <, ª=, ª>,ª<, \=, \> and \<

Relational symbols for evaluation of expressions;

CONTAINS

Can be used to search for the specified value on a virtual terminal screen or in a variable field; and may be qualified by the optional adjectives, PREFIX, WORD or SUFFIX;

IS and IS NOT

Can be used to validate numeric data; and may be qualified by the optional adjectives, DECIMAL, PACKED or ZONED;

AND and OR,

Keywords which further define relationships between expressions for evaluation; the relationships may be parenthesized to control the order in which they are evaluated to determine the truth of a set of conditions;

THEN

A keyword which identifies that the statements which follow are to be executed when the conditional expression is true;

true-routine

Any set of unlabeled Script Language statements to be executed when the conditional expression is true;

NEXT

A keyword which specifies a null operation, and that execution should resume at the next statement following the current IF construct;

ELSE

A keyword which identifies that the statements which follow are to be executed when the conditional expression is false;

false-routine

Any set of unlabeled Script Language statements to be executed when the conditional expression is false;

ENDIF

Delimits the IF routine.

Standard Form Example

```
if &switch = 'y'
then
   perform yes-routine
   set &switch to 'n'
else   perform no-routine   set &switch to 'y'
endif
```

An IF statement must specify one or more conditions. A condition consists of two data items that are connected by one of the relational mnemonics, EQ,NE, GT, GE, LE,LT; the keyword, CONTAINS; or either the keyword, IS or the keywords, IS NOT. One or a logical pair of the relational symbols, =, >, <, ª (not), or \ (not), may be used as an alternative to relational mnemonics. Each of the pair of related data items may be a CA Enterprise Tester screen or variable field expression, a literal, or a number.

When one of a pair of compared arguments is an actual decimal number or numeric variable, and the value of the other argument is a decimal number, then the arguments are compared algebraically, otherwise the arguments are treated as character strings and corresponding characters are compared from left to right based on the EBCDIC collating sequence of the compared characters. When character strings are compared, if the arguments are not the same length, then the shorter of the arguments is considered to be padded with a sufficient number of blanks to make the fields be equal in length. Refer to Numbers for a precise description of decimal numbers, and to Variable Memory for a precise description of numeric and character variables.

When character strings are compared and are found to be unequal, then the variable field, PTENE is set to the number of the first position within the compared fields that is not equal, and the variable field, PTENEL is set to the net length of the longest of the fields being compared, beginning at the first unequal position. When character strings are found to be equal, PTENE and PTENEL are set to zero. When more than a single pair of character strings, connected by the keywords, OR or AND, are compared, then the values of PTENE and PTENEL reflect the results of the last comparison made during evaluation of the specified conditions.

The CONTAINS keyword may be used to search for a specified value on a virtual terminal screen or in a variable field. The CONTAINS condition is true when the argument being searched for is found in the specified screen or variable field. The search conducted when the CONTAINS keyword is used is able to find a desired value even when the alphabetic case of the argument being sought differs from the case in the searched field. The search sees all screen attributes, binary zeros and data in non-display screen fields as blanks, allowing screen search arguments to be based on what is seen on a screen.

When the CONTAINS keyword is used, the data item that follows is the argument being sought. The data item may be preceded by one of the adjectives PREFIX, WORD or SUFFIX to further define the value being sought. When PREFIX is used, only an occurrence of the search argument that is not preceded by, but is followed by an alphabetic or numeric character will be found. When WORD is used, only an occurrence of the search argument that is neither preceded, nor followed by an alphabetic or numeric character will be found. When SUFFIX is used, only an occurrence of the search argument that is preceded by, but is not followed by an alphabetic or numeric character will be found.

The CONTAINS condition provides feedback in variable fields. When a screen field is searched, variable fields PTEARGR and PTEARGC are set to the screen row and column at which the argument is found and PTENETR and PTENETC are set to the screen row and column of any net data, contained in the searched screen field, that follows the argument found. When a variable field is searched, variable field PTEARGI is set to the position within the variable field at which the argument is found and PTENETI is set to the position in the variable field of any net data contained in the searched variable field that follows the argument found. If an argument is found in any source field and more data follows the argument in the searched field, then PTENETL is set to the length of the net field data. When an argument is not found, all applicable feedback variables are set to zero. When no net data follows an argument in a searched field, all applicable net data fields are set to zero.

For example:

```
if <1 1 80> CONTAINS 'ispf/pdf primary option menu'
then
   log 'the panel title begins in screen column ' &PTEARGC
else
   log 'the panel title is not displayed on row 1'
endif
```

When one of the fields being compared is a TRACKED screen field that was read from a CA Enterprise Tester trace data set using the TRACK statement, unequal data may be considered equal if it is described by an active MASK statement. Refer to MASK for details.

The IS or the IS NOT condition may be specified to validate numeric data. The condition must be qualified by one of the adjectives, DECIMAL, PACKED, or ZONED.

For example:

```
if &record(,30,8) is packed
then
   set &decimal-number from packed &record(,30,8)
   log 'value is ' &decimal-number
endif
```

When more than one condition is specified, adjacent conditions are connected by one of the keywords, AND or OR. Conditions connected by AND must both be true for the pair of conditions to be considered true. A pair of conditions connected by OR are considered true if either of the conditions is true. By default, when three or more conditions are specified, consecutive conditions connected by AND must all be true. OR may be used to specify an alternative set of conditions. Parenthesis, (and), may be used to override the default order of evaluating multiple conditions.

For example:

```
if  &v1 = 'a' and &v2 = 'b'  or &v3 = 'c'
```

is equivalent to:

```
if (&v1 = 'a' and &v2 = 'b') or &v3 = 'c'
```

The following statement overrides the default order of evaluation:

```
if &v1 = 'a' and (&v2 = 'b' or &v3 = 'c')
```

The routine to be executed when the conditions are true is identified by a keyword, THEN, which is a required keyword that must immediately follow the IF statement condition(s). The routine must be delimited by an ELSE or ENDIF keyword. A keyword, NEXT, may be specified immediately following THEN to specify that no routine is to be executed when the conditions are true.

The routine to be executed when the conditions are false is identified by a keyword, ELSE, which is optional. The ELSE keyword also serves to identify the end of a THEN routine. An ELSE routine must be delimited by an ENDIF keyword. If an ELSE routine is not specified and IF statement conditions are false, script execution is continued at the statement that follows the ENDIF keyword.

The alternative routines that are executed depending on the truth of the specified conditions may consist of any set of unlabeled Script Language statements, including additional IF statements that may be nested to any desired number of levels. When an ELSE, ENDIF or NEXT keyword is encountered while executing one of the alternative IF routines, script execution is continued at the statement following the ENDIF keyword.

The original CA Enterprise Tester Script Language IF statement did not support the structured IF/THEN/ELSE/ENDIF logic now supported by the Standard Form IF statement described above. The original IF statement format and rules will continue to be supported in this and future releases of CA Enterprise Tester so that CA Enterprise Tester scripts written using Script Language rules supported by previous releases of CA Enterprise Tester will function correctly when processed by current and future releases of CA Enterprise Tester. The original form of the IF statement is now referred to as the Compatibility Form.

The Compatibility Form IF statement evaluates one or more conditions in a manner identical to the Standard Form IF statement. If the conditions are true, the single statement following the IF condition(s) is executed. The statement must be a GOTO, PERFORM, LINK or CALL statement. If the conditions are false, script execution is continued at the statement following the GOTO, PERFORM, LINK or CALL statement.

Compatibility Form Example:

```
if <5 5 *> ne 'SUCCESSFUL'
   go to abnormal-response
```

Compatibility Format:

```
IF first-expression
  EQ | NE |
  GT | GE | LE |LT |
  = |  > |  < |
  ª= | ª> | ª< |
  \= | \> | \< |
  CONTAINS (PREFIX | WORD | SUFFIX) |
      second-expression |
  IS | IS NOT DECIMAL | PACKED | ZONED
 (AND | OR other-expressions)
  GOTO | PERFORM | CALL | LINK
      true-routine
```

first-expression, second-expression and other-expressions are optional expressions for evaluation based on either the relational mnemonics or the relational symbols specified; all conditional expressions supported by the Standard Form IF are also supported by the Compatibility Form IF statement;

true-routine is the target of a single GOTO, PERFORM, LINK or CALL statement that is to be executed when the specified conditions are true.

INVITE

Normally, the INVITE statement checks to see if an online system is attempting to send a message to the CA Enterprise Tester virtual terminal being used for a session, and if a message is available, it is received and placed into the virtual terminal screen buffer. The INVITE statement is also used to receive subsequent messages or to resume processing session events when HANDLE OUTBOUND or HANDLE TIMEOUT statements are used. When HANDLE RESPONSES ON is active, INVITE serves only to set or reset the timer used while receiving a variable number of consecutive messages that leave a session in the contention state. This statement has the following format:

```
INVITE hhmmssth
```

hhmmssth

An optional time interval, expressed in hours (hh), minutes (mm), seconds (ss) and tenths and hundredths of seconds (th), that specifies the maximum amount of time that the INVITE statement should wait for an outbound message. An INVITE time interval may also be expressed via a variable field that contains a value of the form, hhmmssth.

When a Script Language TYPE statement instructs CA Enterprise Tester to send a message to an online system, CA Enterprise Tester sends the message, then waits for and receives the online system response. While waiting for the response, if a message is received that is not a response to the message that was sent, but rather some other message that was sent by the online system, then CA Enterprise Tester rejects the message and waits for the response to the last virtual terminal input. This action simulates the behavior of a real terminal in dealing with messages that are sent by an online system after the online system has sent a message that frees the terminal keyboard and after the terminal operator has begun to key input data.

On a real terminal, however, if a terminal operator has not begun to key data prior to the online system sending another message, then the real terminal would accept and display the message. Because a human terminal operator would always wait for some amount of time for an expected response, a person may not be aware that a terminal keyboard has been freed, and a delayed online system message may appear to be the response to the previous terminal input. The normal operation of a session of this type depends on timing and operator patience.

The execution of a CA Enterprise Tester script is resumed immediately when an online system message has freed the virtual terminal keyboard. If the script is to receive a delayed online system message, then the script must recognize that it has not received the desired response and must provide logic to wait for and receive an anticipated and delayed message.

The Script Language INVITE statement may be used by a script to receive anticipated messages from an online system that are not an immediate response to a terminal input sent by a TYPE statement.

A script that executes a HANDLE OUTBOUND statement in order to gain access to all messages sent by an online system, including messages that do not free the terminal keyboard, must use the INVITE statement to receive subsequent outbound messages while the session is in the OUTBOUND state.

A script that executes a HANDLE TIMEOUT statement may use the INVITE statement, after a session has timed out, to resume the wait for the completion of a timed out session event.

An INVITE statement may specify a maximum amount of time that the INVITE statement should wait for an anticipated outbound message. The INVITE statement will complete whenever an outbound message is received, when a session is passed to another online system during INVITE processing, when session traffic is stopped and restarted by the online system, or when the specified INVITE time interval has elapsed, whichever occurs first.

At the completion of an INVITE statement, the CA Enterprise Tester variable field, PTECC, is set to a count of messages that were received from the online system. A value of zero indicates that a message was not received during INVITE statement processing.

If an INVITE statement completes because the session was passed to another online system or because the online system stopped, then restarted session traffic during INVITE processing, then the variable field, PTESTATE is set to ONLINE to identify the session event.

The HANDLE CONTENTION statement provides other options that may be used by a script to deal with online system messages that are not an immediate responses to terminal inputs caused by the Script Language TYPE statement.

LIMIT

The LIMIT statement permits a script to specify a maximum number of times any single script statement may be executed. This statement has the following format:

```
LIMIT n
```

n

The maximum number of times any script statement may be executed.

Because a CA Enterprise Tester Script has the potential to loop, a LIMIT control is provided. If a LIMIT statement is in effect, a script will be abended if any script statement is executed more times than the limit. By default script statement executions are not limited. If a LIMIT statement is specified, the limit may be subsequently turned off by setting LIMIT to zero.

LINK

The LINK statement allows a script to execute a script subroutine that is a separate member of a CA Enterprise Tester script library.

The LINK statement functions much like the PERFORM statement. When a LINK statement is executed, the location of the script statement that follows the LINK statement is added to a return stack before continuing script execution at the first statement of the named script library member. When an EXIT statement is subsequently executed, script execution is resumed at the statement following the most recently executed LINK or PERFORM statement, after deleting the most recently added return stack entry. This statement has the following format:

```
LINK scriptname
```

Scriptname

The name of a member of a CA Enterprise Tester script library. scriptname may be an actual or literal script name or may be a variable field that contains the name of a script library member.

Searching Script Libraries

When a script specified by the LINK is loaded, available script libraries are searched for the script in the following order:

1. Any script libraries allocated by the PTELIB DD statement in a batch job or the Play Panel link libraries.
2. Any script library referred to by the PTEIN DD statement in batch job, or the Play Panel input script dataset if the dataset is a library.
3. The CA Enterprise Tester system script library specified by the PTEOM ANALYZE,PTELIB=dsn statement in the CA Enterprise Tester system options table. The system script library is displayed on the CA Enterprise Tester Interactive System Analyze Panel.

Messages logged when a script is loaded document the data set name of the library from which the script was loaded.

When a script named by a LINK statement is loaded, the script is unpacked if it was stored in the script library using the ISPF packed data format and is concatenated to the main script or other linked scripts. An EXIT statement is always added to the end of a script that is loaded to resolve a LINK statement.

If multiple LINK statements refer to the same script library member, then only a single copy of the script is loaded and each LINK statement that refers to the script, refers to the single copy of the script.

A script that is loaded by a LINK statement acts as a logical extension of the script containing the LINK statement. A script that contains a LINK statement and the script that is named by a LINK statement share a common variable pool and CA Enterprise Tester session. Script labels are also shared. In fact, a label contained in a script that is loaded by a LINK statement may be referred to by GOTO, PERFORM or other statements contained in the main script. This also means that a linking and linked-to scripts must contain unique script labels. An appropriate script label naming convention may be used to avoid the possibility of having duplicate labels in linking and linked-to scripts. Because a script named by a LINK statement is loaded, if possible, while the main script is being loaded for execution, a LINK statement may be used to copy other routines that may be used by the main script, even if the LINK statement that named the script is never executed.

While loading a script, if a LINK statement is found that specifies a script name via a variable field and if the variable field does not contain a script name, then a script is not loaded. When a script is being executed, if a LINK statement is found that specifies a script name via a variable field and if the variable field does not contain a script name, the LINK statement is tolerated and no action is performed. As a result, a script may be designed to LINK to a subroutine only when a value is assigned to the variable, possibly by assigning a value to the LINKvariable via the JCL EXEC PARM field or via the Play Panel parameter field. Additionally, a LINK statement may be turned off during script execution by setting the LINKvariable to a null value.

When a script is being executed, if a LINK statement is found that specifies a script name via a variable field and if the variable field contains the name of a script that has not been previously loaded, then the named script is dynamically loaded before being performed in the normal manner. By changing the value of a variable specified in a LINK statement, it is possible for a LINKstatement to perform a different script every time it is executed.

Scripts that are loaded to resolve a LINK statement may contain additional LINK statements. Scripts named by LINKstatements are always loaded at the earliest possible opportunity, either while a script is being loaded for execution or when a LINK statement is executed that identifies a script that has not been previously loaded.

LOG

The LOG statement is used to write a message to the CA Enterprise Tester Log. When a script is executed as a batch job step, the CA Enterprise Tester Log is the data set specified on the PTELOG DD statement. For a script executed via the Interactive System PLAY command, the CA Enterprise Tester Log is the scrollable Message Log, accessible via the Interactive System Monitor Panel. This statement has the following format:

```
LOG string
```

string

A free-form list of data items that represents the message to be logged.

Example

```
LOG 'TRANSACTION ' &TRAN
   ' RETURNED MESSAGE: ' <24 1 40>
```

The string of data items specified on a LOG statement may include any combination of literals, screen fields and variable field expressions.

When the LOG statement is executed, the message to be logged is constructed by concatenating the literal data with the contents of the screen and variable fields specified. The message is then time-stamped and written to the log.

LOGOFF

The LOGOFF statement causes orderly termination of an active session with an online system. This statement has the following format:

LOGOFF

When a script is executed as a batch job step, a LOGOFF is executed automatically as a part of END, ABEND, or LOGON statement processing. When a script containing a LOGON statement is played using the CA Enterprise Tester Interactive System, the execution of the LOGONstatement first invokes LOGOFF processing if a session is already active on the Interactive System port on which the script is played.

LOGON

The LOGON statement starts a session between a CA Enterprise Tester virtual terminal and a named online system. This statement has the following format:

```
LOGON system-name
   ( MODEL terminal-model
     MOD2 | MOD3 | MOD4 | MOD5 )
   ( default-screen-size
     24X80 | 32X80 | 43X80 | 27X132 )
   ( alternate-screen-size
     24X80 | 32X80 | 43X80 | 27X132 )
   ( terminal-type
     BASICDS | EXTDS )
   ( FROM terminal-name )
   ( DATA logon-data )
```

system-name

The one through eight-character name of an online system with which a session is to be started, and is required data. The system name specified may be an actual name, the literal value or a variable field that contains the name of an online system.

MODEL

An optional keyword and indicates that the data which follows is a symbolic terminal model name or the name of a mode table entry.

terminal-model

A data item identified by the MODEL keyword, and is the logon mode table entry to be used to start a session.

terminal-model

Can be one of the keywords MOD2, MOD3, MOD4or MOD5, to refer symbolically to actual mode table entry names defined at an installation, or may be the actual name of a logon mode table entry. terminal-model may be expressed as an actual value, or may be a variable field that contains a valid terminal model value or a null value.

default-screen-size

Optional, and is the default virtual terminal screen size to be used for a session. default-screen-size may be one of the keywords 24X80, 32X80, 43X80or 27X132, and may be expressed as an actual value, or may be a variable field that contains a screen size keyword or a null value.

alternate-screen-size

Optional, and is the alternate virtual terminal screen size to be used for a session. alternate-screen-size may be one of the keywords 24X80, 32X80, 43X80or 27X132; and may be expressed as an actual value, or may be a variable field that contains a screen size keyword or a null value.

terminal-type

Optional, and specifies the type of terminal to be used for a session. terminal-type may be one of the keywords BASICDS or EXTDS, and may be expressed as an actual value, or may be a variable field that contains a terminal type keyword or null value.

FROM

An optional keyword and indicates that the data item that follows is a virtual terminal name.

terminal-name

A data item identified by the FROM keyword, and is the name of the virtual terminal to be used for the session.

terminal-name

Can be the actual name of a VTAM APPL that has been defined at an installation for use as a CA Enterprise Tester virtual terminal or may be a variable field that contains a virtual terminal name or a null value. If the FROM keyword is not specified or if terminal-name is a variable field that contains a null value, then a virtual terminal will be assigned automatically.

DATA

An optional keyword and indicates that the data item that follows is logon data.

logon-data

A data item or items identified by the DATA keyword, and is logon data that is to be sent to an online system with a session request. logon-data may be a variable field that contains logon data and may be up to 255 characters in length; or may be one or more consecutive literal values whose accumulated length(s) do not exceed 255 characters. When the DATA keyword is not specified or if logon-data is a variable field that contains no data, then data is not included in the session request sent to an online system.

Examples

```
LOGON TSO
MODEL D4A32XX3 24X80 43X80
EXTDS
DATA "RS2 PROC(TSORS1T)"
LOGON &system
      MODEL &terminal-model
      &default-size
      &alternate-size
      &terminal-type
      DATA &logon-data
LOGON CICS1
LOGON CICS2 FROM PT99
      MODEL MOD2 BASICDS
```

The system name specified in a LOGON statement may be the actual VTAM APPL name used by an online system to connect to VTAM or may be a familiar name that has been associated with an actual VTAM APPL name during CA Enterprise Tester installation. Refer to the CA Enterprise Tester System Guide for information on how to establish familiar names for online systems.

If the MODEL keyword is not specified, a suitable mode table entry will be selected based on other LOGON statement parameters and system defaults. The logon mode table entry used to start a session contains session parameters and may contain screen sizes and other device related information. Mode table entries that contain screen sizes may cause a session to be started that uses screen sizes other than those specified using other LOGON statement parameters.

The first screen size keyword encountered in a LOGON statement specifies the default screen size. If a default screen size is specified, it is used only when a session is started using a mode table entry that does not contain screen sizes and sizes are not forced by an online system.

The second screen size keyword encountered in a LOGON statement specifies the alternate screen size. If an alternate screen size is specified, it is used only when a session is started using a mode table entry that does not contain screen sizes and sizes are not forced by an online system. If screen sizes are not specified, are not defined in the mode table used to start a session and are not forced by an online system, then default sizes specified at an installation are used.

A terminal may be either a basic or extended 3270 data stream terminal. A basic data stream terminal has a single screen buffer that is used to store screen data and coded basic field attributes. An extended data stream terminal has a screen buffer that is used to store screen data and actual basic field attributes and has an extended attribute buffer that is used to store extended data stream color and highlighting attributes and other information. 3270 Attribute Values documents the exact information that is stored in either basic or extended data stream terminal buffers.

If terminal type is not specified, a default type specified at an installation is used. If an online system sends a 3270 Read Partition Query structured field to a terminal to determine device capabilities, the terminal type specified influences the contents of the reply.

Session Control shows how the LOGON statement and other statements may be used to start and control CA Enterprise Tester sessions and provides additional information about LOGON statement parameters.

MASK

The MASK statement is used to suppress the comparison of specified screen data that was read from a CA Enterprise Tester trace data set using the TRACK statement, when the tracked screen image is compared to other data.

```
MASK specifications
     ( ON | OFF | RESET )
```

specifications

Can include a data picture and/or a screen location that is to be masked when TRACKED screen data is compared to other data; and may include one of the following optional keywords:

- ON activates a previously inactivated list of masks;
- OFF inactivates the masking feature and remembers previously specified masks;
- RESET inactivates the masking feature and purges previously specified masks.

Example

```
MASK 'ZN-NN-NN' <1,67>
MASK 'ZN:NN' <1,76>
MASK 'NN/NN/NN'
MASK <24>
```

When a screen image is read from a CA Enterprise Tester trace data set by executing a Script Language TRACK statement, the screen that was read may be referred to by the script using a standard CA Enterprise Tester screen field expression that is preceded by the TRACKED adjective.

For example:

```
IF <&R 1 &PTECOLS> EQ TRACKED <&R 1 &PTETCOLS>   GO TO NEXTROW
```

In the previous example, a row of the active virtual terminal screen is compared to a row of a tracked screen and if the compared screen rows are equal, script execution resumes at the script label, NEXTROW. When tracked screen data is compared to other data, the comparison is influenced by previously executed MASK statements. For example, if a transaction displays current time, a MASK statement may be used to cause different times to be considered equal when a live screen image is compared to a tracked screen image.

The following MASK statement causes times displayed at row 1, column 73, to be considered equal, provided that the actual times displayed on the compared screen images conform to the specified picture:

```
MASK <1 73> 'nn:nn:nn'
```

The primary purpose in comparing active session screen data to traced screen data is in a regression testing application. CA Enterprise Tester provides two general purpose session analyzers that may be used to perform online application regression testing. A session analyzer is a CA Enterprise Tester script that re-creates traced sessions and compares live online system responses to traced responses to detect differences. Refer to The CA Enterprise Tester Session Analyzer for a description of the distributed CA Enterprise Tester session analyzers.

The MASK statement may be used to customize session analyzer processing to automatically suppress the comparison of predictable and insignificant screen field mismatches such as date and time, to more fully automate and to speed the regression testing process. Masking Dates, Times, and Other Data shows how to use the MASK statement to customize the comparison of sessions that execute your online applications.

A MASK picture is a string of characters that represents the format of a screen field. A picture consists of a string of pattern characters that define the data that is valid in each position of a string of data. A field of data is considered to conform to a picture only when every position of the string contains data defined by the corresponding character of the picture. A picture may contain the following pattern characters:

- Z—Corresponding field character must be blank or numeric
- N—Corresponding field character must be numeric
- A—Corresponding character must be alphabetic, blank or a hyphen
- X—Corresponding character may be anything
- A special symbol or a blank - if the previous pattern character is Z, and if the field character that corresponds to the Z is blank, then the field character that corresponds to the special symbol or blank pattern character may be blank, otherwise, the field character that corresponds to a special symbol or blank in the pattern must be equal to the pattern character.

When a MASK statement that specifies a picture and/or a screen location is executed, the masking feature is activated and the specified mask is added to a queue of masks. Masks are added to the queue in descending sequence according to the length of the mask. Masks of the same length are added to the queue in the order in which the MASK statements are executed.

When tracked data is compared to other data, the tracked data is examined from left to right to identify all masks that appear in the data. The first mask that matches the left-most masked character in the tracked data masks all of the data characters that correspond to the mask, provided that, for picture masks, the data being compared to the tracked data also conforms to the picture. When masked data is identified, the data preceding the masked data is compared, and if unequal, determines the result of the comparison. If the data preceding masked data is equal, the masked data is also considered equal, and the data following masked data is examined to identify additional masked data.

The following list documents all supported combinations of MASK statement specifications. Row, column and field length values may be expressed as absolute numbers or as numeric variable fields that contain valid row, column or length values.

- MASK <r> unconditionally masks row r.
- MASK <r,c> unconditionally masks row r from column cto the end of the row.
- MASK <r,c,l> unconditionally masks the field at row r and column c. The field length is l. The field specified must be wholly contained within a single screen row.

- MASK <,c> unconditionally masks screen columns beginning at column c and ending at the end of each row.
- MASK <,c,l> unconditionally masks the screen fields that begin at column c of each screen row and whose lengths are l. The field specified must be wholly contained within a single screen row.
- MASK picture <r,c> masks the screen field at row r and column c if the field data conforms to the specified picture. The field specified must be wholly contained within a single screen row.
- MASK picture <r> masks the screen data on row r that conforms to the specified picture.
- MASK picture <,c> masks screen data at column c that conforms to the specified picture. The field specified must be wholly contained within a single screen row.
- MASK picture masks all screen data that conforms to the specified picture.
- MASK OFF inactivates the masking feature, but remembers previously specified masks for future activation.
- MASK ON activates a previously inactivated list of masks.
- MASK RESET inactivates the masking feature and purges all all previously specified masks.

For all MASK statements listed above, r may be specified as r1-rn, where r1 is the first, and rn is the last screen row to which the MASK applies.

For example, the following statement masks screen rows 3 through 20:

```
MASK <3-20>
```

Because the through symbol, (a hyphen, -), is also a valid variable field name character, if a variable value is used as r1, then at least one blank should separate the variable name and the through symbol.

For example:

```
MASK <&row-1 - &row-n>
```

NEXT

The NEXT keyword may be used in an IF statement to specify a null operation, and that execution should resume at the next statement following the current IF construct. Refer to IF for details.

OPEN

The OPEN statement prepares a user data set for subsequent processing. The data set may be a fixed or variable length sequential data set, a member of a PDS or GDG, or a VSAM KSDS. An OPEN statement must be executed for a data set prior to executing other data set access statements.

Examples

```
OPEN INPUT INDD ENDFILE EOFPROC

OPEN OUTPUT OUTDD
```

Format

```
OPEN INPUT | OUTPUT ddname
  ( UNPACK )
  ( WARNING routine )
  ( ENDFILE routine )
  ( ERROR routine | * )
```

INPUT

Specifies that the data set is to be opened for read-only access;

OUTPUT

Specifies that the data set being opened will be written to or otherwise modified;

ddname

The name of a data definition, (DD), statement that identifies the data set to be opened; ddname may be the actual DD name or may be expressed as a variable field that contains the DD name; the DDname specified may not begin with the characters, PTE;

UNPACK

Specifies that if the data set being opened is packed, (using ISPF packed data format), then subsequent READ statements are to return unpacked, logical records.

CA Enterprise Tester provides automatic unusual condition processing when an optional condition keyword is specified. Each condition keyword is followed by the name of a script routine that is to receive control if the condition is encountered.

A condition handling routine may be expressed as a statement label or number, as a variable field that contains a statement label or number, or may be the special symbol, asterisk, *, to specify that execution is to be resumed at the next script statement. Condition keywords supported by the OPEN statement are WARNING, ENDFILE and ERROR.

WARNING routine

Identifies a script routine that is to receive control when a VSAM open warning occurs. If WARNING is not specified, a VSAM open warning is considered to be an error.

ENDFILE routine

Identifies a script routine that is to receive control when an end-of-file condition is encountered during the execution of a READ statement; the ENDFILE keyword may optionally be specified on the READ statement.

ERROR routine

Identifies a script routine that is to receive control when an error or other unhandled condition is encountered during open or subsequent data set access; the ERROR keyword may also be specified on other data set access statements.

If a VSAM data set is opened and a warning is issued by VSAM, the data set will be opened for subsequent access only if the warning is handled by specifying the WARNING keyword.

OPEN statement ENDFILE and ERROR handling specifications are remembered. If a handled condition is encountered during the execution of a subsequent READ, TRACK, WRITE, UPDATE, DELETE or CLOSE statement that does not also specify handling for the condition, then OPEN statement condition handling specifications will be used.

If the execution of a data set access statement does not complete normally, and no condition handling has been specified, then the error is handled according to the default or currently active HANDLE statement ILLOGIC specification.

During the execution of every CA Enterprise Tester data set access statement, the variable field, PTERESP, is set to indicate the success of the request. An error handling routine may examine PTERESP to determine the exact cause of an error. Refer to PTERESP in CA Enterprise Tester Defined Variables for a description of data set access response codes and particular error codes.

During the execution of an OPEN statement, data set organization, record format and record length specifications are obtained from standard data set labels, JCL DD statement DCBRECFM, LRECL and BLKSIZE specifications, TSO ALLOC command parameters or from the VSAM catalog. This information is used to determine the length of the variable data created during the execution of subsequent READ statements, and to control the construction of records created during the execution of WRITE and UPDATE statements.

The OPEN statement keyword, UNPACK, specifies that if the opened dataset was stored using the ISPF packed data format, such as a trace data set created by the CA Enterprise Tester Trace Facility, then a subsequent READ statement should return unpacked logical records. When an OPEN statement refers to a DD statement that is concatenated data, and the OPEN statement specifies UNPACK, the concatenated datasets may be a mixture of packed and unpacked datasets.

UNPACK may be specified only for RECFM=F datasets with LRECL=256 or less. The OPEN statement fails if UNPACK is specified for other types of datasets.

Refer to Data Set Access for an overview level description of CA Enterprise Tester data set access capabilities.

OR

The OR keyword is used to join conditions in IF and DO statements, and specifies that the conditions are true if either condition is true. Refer to IF and DO for details.

PAUSE

The PAUSE statement allows a script to stop and wait for action by a CA Enterprise Tester Interactive System terminal user. The PAUSE statement has the following format:

```
PAUSE
```

The PAUSE statement operates only when a script is executing under the control of the CA Enterprise Tester Interactive System. Execution of a PAUSE statement stops script execution and causes the script to wait for action by the Interactive System terminal user. While at a PAUSE the terminal user may enter Interactive System END, EXECUTE, RESUME, STEP or WAIT commands to effect subsequent script execution.

After logging instructions or messages, the PAUSE statement may be used to stop the script and wait for terminal user action.

PERFORM

The PERFORM statement allows a script to execute a subroutine that is located at some other point within the script. This statement has the following format:

```
PERFORM routine
```

routine

Identifies the script subroutine to be executed. routine may be a script label or statement number or may be a variable field that contains a script label or statement number.

When a PERFORM statement is executed, the location of the script statement that follows the PERFORM statement is added to a return stack before continuing script execution at the identified script subroutine. When an EXIT statement is subsequently executed, script execution is resumed at the statement following the most recently executed LINK or PERFORM statement, after deleting the most recently added return stack entry.

Performed routines may PERFORM other routines to any number of levels.

READ

The READ statement is used to read the next record from a sequential data set, to sequentially read records in a VSAM Key Sequenced Data Set, (KSDS), in a forward or backward direction, or to randomly read a record in a VSAM KSDS that is identified by a full or generic key. The record is placed in a variable field for use by the script. This statement has the following format:

```
READ ddname INTO record
    ( KEY value )
    ( NEXT | PREVIOUS )
    ( NOTFOUND routine )
    ( ENDFILE routine )
    ( ERROR routine | * )
```

ddname

The name of a data definition, (DD), statement that identifies the data set to be read; ddname may be the actual DDname or may be a variable field that contains the DD name; ddname must have been previously opened by executing an OPEN statement; the DD name specified may not begin with the characters, PTE;

INTO

A required keyword and indicates that a variable field expression follows:

- record is a variable field into which the record is to be read;
- KEY is an optional keyword that specifies that a keyed read is to be performed and that a literal or variable key value follows; KEY is valid only when processing a VSAM KSDS;
- value is a literal key value or a variable field that contains the key of a record to be read; the key value may be a generic or full key value;

Examples:

```
READ INDD   INTO &REC ENDFILE EOF
READ VSAMDD INTO &REC KEY &KEY NOTFOUND NOTFND
READ VSAMDD INTO &REC PREVIOUS ENDFILE EOF
```

One of the optional keywords, NEXT or PREVIOUS, specifies the direction in which a VSAM KSDS is to be read; if direction is not specified, NEXT is assumed;

CA Enterprise Tester provides automatic unusual condition processing when an optional condition keyword is specified. Each condition keyword is followed by the name of a script routine that is to receive control if the condition is encountered.

A condition handling routine may be expressed as a statement label or number, as a variable field that contains a statement label or number, or may be the special symbol, asterisk, *, to specify that execution is to be resumed at the next script statement. Condition keywords supported by the READ statement are NOTFOUND, ENDFILE and ERROR.

- NOTFOUND routine identifies a script routine that is to receive control when a key value specified by the current READ statement is not found in a VSAM KSDS or when a record read sequentially does not contain the generic key specified by the most recently executed keyed read. Coding NOTFOUND specifies that only a record matching the current or most recently specified KEY can satisfy the READ request.
- ENDFILE routine identifies a script routine that is to receive control when an end-of-file condition is encountered during the execution of a READ statement. If ENDFILE is not specified on a READ statement, an ENDFILE routine specified on an OPEN statement will be used. If ENDFILE is not specified and the condition occurs, the end-of-file condition is considered to be an ERROR.
- ERROR routine identifies a script routine that is to receive control when an error or other unhandled condition is encountered during the execution of a READ statement. IF ERROR is not specified on a READ statement, an ERROR routine specified on the associated OPEN statement will be used.

The READ statement places the record read into the variable field identified by the INTO parameter. A record may be read into a variable field, an entry of a variable array or into a substring of a character variable field, by coding the appropriate variable occurrence and/or substring specifications. If the variable into which the data is being read does not specify a substring length then the length of the variable is the length of the logical record read. If a substring length is specified, and the length differs from the length of the record read, then the record is truncated or the target substring is padded with blanks as required. When a variable length record has been read, the length of the record may be obtained as follows:

```
READ VLDD INTO &RECORDSET &LENGTH TO LENGTH &RECORD
```

The INTO character variable field that is created or modified by the execution of a READ statement may be subsequently used by a script in any way that other variable memory may be used.

A sequential data set may be read only sequentially and only in a forward direction. The keywords KEY, PREVIOUS, and NOTFOUND are not permitted for READ statements used to access sequential data sets.

A VSAM KSDS may be read sequentially in a forward or backward direction or may be read randomly. A sequential read is requested whenever a KEY is not specified. While processing a VSAM KSDS, the data set is considered to be positioned at the record most recently read. When the NEXT keyword is specified or PREVIOUS is not specified, a sequential read request will get the first record that follows the current data set position. If a record has not been previously read, a read NEXT will get the first record in the data set.

When PREVIOUS is specified, a sequential read will get the first record that precedes the current data set position. If a record has not been previously read, a READ PREVIOUS will get the last record in the data set.

If the NOTFOUND keyword is specified for a sequential read in either direction, the READ statement will get a record only when the key of the next or previous record begins with the generic key value specified by the most recently executed READ statement that specified a KEY.

Sequential read position is lost when a READ statement encounters a NOTFOUND, ENDFILE or ERROR condition. A subsequent READ statement must specify a KEY to establish a new data set position.

A VSAM KSDS is read randomly when the KEY keyword and a key value is specified. When a keyed read request specifies the NEXT keyword, or if PREVIOUS is not specified, then the read request will get the first record in the data set whose key value is greater than or equal to the full or generic key value specified. A keyed READ request that specifies the PREVIOUS keyword, will get the last record in the data set whose key begins with a value that is less than or equal to the full or generic key value specified. If the NOTFOUND keyword is specified, a keyed read request will get a record only when the full or generic key specified exactly matches the key of a record in the VSAM KSDS.

During the execution of the READ statement, the variable field, PTERESP, is set to indicate the success of the request. An error handling routine may examine PTERESP to determine the exact cause of an error. Refer to PTERESP in CA Enterprise Tester Defined Variables for a description of data set access response codes and particular error codes.

Refer to Data Set Access for an overview level description of CA Enterprise Tester data set access capabilities.

RECORD

If a session recording is started using the Interactive System RECORD command, then session activity generated by a script or Interactive System user is captured in a specified recording data set in the form of a CA Enterprise Tester script or REXX exec. The generated script contains LOGON, TYPE, DELAY, INVITE and LOGOFF statements that capture session events that occur while the recording is active. Session activity may also be recorded from a CA Enterprise Tester script executed as a batch job.

When a PTEREC DD statement is present in CA Enterprise Tester Batch System JCL, a session recording will automatically be initiated when the script is loaded for execution, will optionally be ended when script execution is ended, and will be stored in the data set specified by the PTEREC DD statement.

The RECORD statement allows a script to selectively record session activity and to control the destination, format and contents of the recorded script. This statement has the following format:

```
RECORD ON | OFF
( PSL | REXX )
( TO ddname ( PASS ) )
( COMMENTS number )
( DELAYS | DELAYSOFF )
( MDTFIELDS | CHANGES )
( PACK | NOPACK )( statement )
```

ON

Specifies subsequent session activity is to be recorded.

OFF

Specifies subsequent session activity should not be recorded;

PSL

Specifies the session recording should be captured in CA Enterprise Tester Script language; REXX specifies the session should be captured in a REXX exec, which may be executed with the CA Enterprise Tester Host Command Environment.

TO ddname

Optional, and specifies the name of the data definition, (DD), to which subsequent session activity is to be written, opens the data set, and initiates the recording function;

PASS

Optional, specifies that the recording is to remain active after a script played via the Interactive System ends.

COMMENTS

Optional, and specifies that screen image lines are to be recorded as comments prior to each recorded TYPE statement.

Number

A literal number specifying how many screen image lines are to be recorded.

DELAYS

Specifies that actual user think time is to be recorded with DELAY statements; DELAYSOFF specifies that DELAY statements should not be recorded.

MDTFIELDS

Specifies whether or not all MDT (Modified Data Tag) fields, including all MDT field data should be recorded; CHANGES specifies only changed fields, excluding unchanged trailing nulls, are to be recorded.

PACK

Optional and specifies that the recorded script is to be written using the ISPF packed data format; NOPACK, specifies that the recorded script is to be written as unpacked, fixed-length records;

statement

An optional list of data items, which may include variable fields, screen fields, literals and numbers whose values are to be concatenated to construct and write a Script Language statement to the recording data set.

Examples

```
RECORD TO DDSCRPT1
RECORD '* AP MENU PROCESSING BEGINS'
```

Default values for all optional parameters are determined by system installation specifications.

RECORD statements that are executed while a PTEREC DD is not present in batch job JCL or that are executed by a script played via the Interactive System when a recording has not been started, are tolerated and perform no operation.

A script may selectively control the recording of session activity.

- By default, if a PTEREC DD statement is included in batch JCL, session activity will be recorded.
- A RECORD OFF statement may be executed to suppress the recording of subsequent session activity. Turning a recording off does not close the recording data set, but rather simply suspends recording of session events until the recording is turned back on.

- When a RECORD ON statement is executed, subsequent session activity will be recorded.
- An ENDRECORD statement ends the recording and closes the data set.
- A single recording data set may be active on a port. If a RECORD TO ddname has been executed, or a recording is active because of a PTEREC DD statement, a subsequent RECORD TO ddname statement will end the active recording before starting the new one.
- PASS may be specified on a RECORD TO statement to specify that the recording is to remain active after a script played via the Interactive System ends. When a script is played interactively, it may start a session, recording and/or trace, and end, leaving the started functions active to capture subsequent interactive session activity entered manually on the session started by the script. This technique may simplify session capture activity by eliminating the need to manually start a session, recording and/or trace.

The RECORD statement may be used to specify the language to use for the recording, and may be one of the keywords, PSL, (ProTerm Script Language), or REXX. The language of the recording may also be specified via the CA Enterprise Tester Interactive System Record Panel when a recording is started or may be specified using the JCL PARM field keywords, PSL or REXX, when a script is executed as a batch job. If a format is not specified, a default, specified during CA Enterprise Tester installation, is used.

- RECORD PSL specifies that the recorded script is to be written in the CA Enterprise Tester Script Language creating a script which may be executed using the Interactive System Play Panel or from a batch job.
- RECORD REXX specifies that the recorded session activity is to be written in REXX, creating a REXX exec which may be executed using the CA Enterprise Tester Host Command Environment which enables REXX to ADDRESS PROTERM and execute CA Enterprise Tester Host Commands. For complete documentation of the CA Enterprise Tester Host Command Environment for REXX, please refer to CA Enterprise Tester Commands for REXX.
- The RECORD statement may be used to specify options which describe data to include in a recorded script and may be one of the keywords, DELAYS or DELAYSOFF, MDTFIELDS or CHANGES. Data to include in the recorded script may also be specified via the Interactive System Record Panel when a recording is started or may be specified using the JCL PARM field keywords, RDELAY or RDELAYOFF, and RMDT or RCHG, when a script is executed as a batch job. If an option is not specified, a default, specified during CA Enterprise Tester installation, is used.

- RECORD DELAYS specifies that the recorded script is to include actual times between transactions that occurred when a live session was captured. By default, scripts that contain generated DELAY statements will execute at the same speed as the captured session. (Or, more precisely, a very close approximation of the same speed, assuming all system variables are similar to those at capture time). However, a DELAY PERCENT may be specified that increases or decreases each DELAY interval by the specified percentage. By specifying an appropriate DELAY percent, a single set of scripts can produce a larger or smaller workload by compressing or expanding the time in which they run.
- RECORD DELAYSOFF specifies that recorded scripts should not capture delay time intervals. For more information on the DELAY statement, refer to DELAY. For more information on managing workloads delivered to a system please refer to Multiple Session Testing.
- RECORD MDTFIELDS specifies that all MDT, (Modified Data Tag), fields, and all data in MDT fields should be captured in a recorded script. Data areas, or fields in an online terminal screen image may be primed by an application to be returned to the application by setting the MDT bit in the field attribute byte. This option instructs CA Enterprise Tester to include this data in the recorded script.
- RECORD CHANGES specifies that only data which is changed by user input should be captured, excluding unchanged trailing nulls.
- The RECORD COMMENT statement may be used to specify that lines from the virtual terminal screen image are to be included in the recorded script as comments.
- The RECORD statement may be used to specify the format of the recorded script and may be one of the keywords, PACK or NOPACK. The format of recorded script may also be specified via the Interactive System Record Panel when a recording is started or may be specified using the JCL PARM field keywords, PRECORD or NOPRECORD, when a script is executed as a batch job. If a format is not specified, a default, specified during installation, is used.
- RECORD PACK specifies that the recorded script is to be written to the record data set using the ISPF packed data format. The packed data format significantly reduces the disk space required for a script. Packed scripts appear to be ordinary data when accessed using ISPF/PDF EDIT or BROWSE or when processed by any CA Enterprise Tester function. Packed scripts may be unpacked using ISPF/PDF EDIT, by saving an edited script after entering the PDF command, PACK OFF.
- RECORD NOPACK specifies that recorded scripts are to be written as normal unpacked fixed length records. Scripts that are not packed may be printed using standard print utilities, including the TSO PRINTOFF command.

The RECORD statement may be used by a script to construct and write Script Language statements to an active recording. For example, a CA Enterprise Tester trace may be converted to a session recording by starting a recording before playing a script that is designed to repeat a traced session. The distributed CA Enterprise Tester session analyzer, PTESA, in the distributed Script library, contains logic to record TRACE statements when captions are found in the session trace data set being analyzed. As a result, if a recording is started prior to using a session analyzer to repeat a session, then the recorded script will contain TRACE statements that echo the screen captions.

The following statement is used to generate the recorded TRACE statements:

```
RECORD ' TRACE "' &PTETCAPT '"'
```

It is also possible to convert a recorded session to a session trace. This may be accomplished by starting a session trace before playing a recorded script. A script that contains TRACE statements that set captions, causes the captions to be included on the traced session screens. These features allow captured sessions to be documented in a way that allows the documentation to be retained when recordings are converted to traces or when traces are converted to recordings.

CA Enterprise Tester also provides utilities which may be used to convert a format trace to a script or REXX exec. For a complete description of CA Enterprise Tester Utilities, please refer to CA Enterprise Tester Utilities.

SET

The SET statement may be used to assign, modify, interrogate, convert, edit or otherwise manipulate the contents of a variable field, an entry in a variable field array, or a substring of a character variable field.

Examples

Assign a value to or modify the value of a variable field, an entry in a variable field array, or a substring of a character variable field.

```
set &switch      TO 'yes'
set &item(&n)    TO <&row &col &len> only
set &previtem    TO &item
```

Propagate values throughout a character variable, a substring of a character variable, or into all entries of a new variable field array.

```
set &pat(,1,80)  TO ALL '-'
set &total(12)   TO 0
```

Concatenate values into a single variable field.

```
set &mm-dd-yy    TO &ptemm '-' &ptedd '-' &pteyy
```

Initialize, increment or decrement a numeric variable.

```
set &index    TO 0
set &index    UP 1
set &count    DOWN 1
```

Convert data from binary, packed or zoned format to decimal format.

```
set &dec-nbr     FROM SIGNED BINARY   &rec(,32,4)
set &dollars     FROM PACKED PRECISION 2 &rec(,40,9)
set &count       FROM ZONED &rec(,52,2)
```

Convert decimal data to binary, packed or zoned format.

```
set &rec(,32,4)  TO SIGNED BINARY   &dec-nbr
set &rec(,40,9)  TO PACKED PRECISION 2 &dollars
set &rec(,52,2)  TO ZONED &count
```

Edit decimal data.

```
set &col2(,1,15) to &number SIGN PUNCTUATE JUSTIFY CENTER
```

Determine the length of a variable field.

```
set &length   to LENGTH &record
```

Transform or convert the form of character data.

```
set &FF          TO HEX 'FF'set &chars-A-F   FROM HEX &data
set &capital     TO CAPITAL <1 1 80>
set &flag-bits   FROM BITS &rec(,40,1)
set &rec(,40,1)  TO BITS '10000111'
```

Create an edited version of virtual terminal screen data that contains blanks in the place of 3270 attributes and non-display screen data, and dashes in place of graphic escape characters.

```
set &prtline     to IMAGE <&row,1,&ptecols>
```

Determine the attributes of a virtual terminal screen location.

```
set &attr1    to ATTRIBUTES <12 25>
set &attr2    to ATTRIBUTES TRACKED <12 25>
```

Format

```
SET targetpreposition
TO | FROM |
UP ( BY ) | DOWN ( BY )
( adjective(s)
ALL |ATTRIBUTES ( TRACKED ) |
BITS |
CAPITAL |
DECIMAL |
HEX |
IMAGE ( TRACKED ) |
LENGTH |
PACKED ( PRECISION n ) |
 ( SIGNED ) BINARY ( PRECISION n ) |
STRING |
ZONED ( PRECISION n ) )
source | *
 ( adverb(s)
( ONLY )
( SIGN )
( PUNCTUATE )
( JUSTIFY LEFT | CENTER | RIGHT ) )
```

target

The target variable field or substring of a variable field to which a value is to be assigned. Unless the value assigned by a SET statement is an unedited decimal value, target becomes or remains a character variable field. When the value assigned by a SET statement is an unedited decimal value, target becomes a numeric variable field. A decimal value is edited when substring notation is used in the target variable field expression or when any of the adverbs, SIGN, PUNCTUATE, or JUSTIFY, are used to edit the decimal value. Refer to Variable Fields for a complete description of the syntax of variable field expressions and substring notation.

preposition

One of the required keywords TO or FROM; or Upon DOWN, optionally with the keyword, BY. All prepositions indicate that SET statement specifications, that describe the value to be assigned to target, follow.

- TO or FROM, when used with one of the adjectives, BINARY, PACKED, ZONED, CAPITAL, HEX, or BITS, specify that a character or decimal source value is to be converted TO the specified data format, or that the source value is to be converted FROM the specified data format to decimal or character format.
- In any other case, TO and FROM, have the same meaning and simply indicate that other SET statement specifications follow.

- Upon UP BY specify that the decimal source value is to be added to the target variable.
- DOWN or DOWN BY specify that the decimal source value is to be subtracted from the target variable.

adjective(s)

Optional keywords that may be specified when TO or FROM are specified. Adjectives specify properties of the source and/or target value, and specify an operation that is to be performed on the source value to obtain the value to be assigned to the target variable.

If an adjective is not specified, and source is a decimal number, a numeric variable, or the special symbol, * , and if the value is not edited by storing the value in a substring of the target variable field or by specifying one of the adverbs, SIGN, PUNCTUATE or JUSTIFY, then the unedited decimal source value is stored in target and target becomes a numeric variable field. In any other case, if an adjective is not specified, then the source data is copied to target, padding the value with blanks if it is shorter than a specified target substring length, or truncating the value if it is longer than a specified target substring length, and target becomes or remains a character variable field.

When an adjective is specified, the following processing occurs:

- ALL propagates the source value into the target variable, as many times as it will fit. Substring position and length may be used to specify the substring of the target variable into which the source value is to be propagated. For example, &target(,1,133), specifies that the source value is to be propagated into the first 133 positions of &target.
- ATTRIBUTES examines the TRACKEDor virtual terminal screen location specified as the source data, and assigns a value to the target variable that consists of a list of keywords that describe all 3270 attributes of the specified screen location. Subsequently, the CONTAINS condition may be used in an IF statement to determine if the target variable contains a keyword describing a particular 3270 attribute, for example, the word MODIFIABLE. Refer to 3270 Attribute Values for a list of all possible 3270 attribute keywords.
- BINARY converts a decimal source value to binary format when TOBINARY is specified, or converts a binary source value to decimal format when FROM BINARY is specified. When BINARY is specified, the adjectives, SIGNED and/or PRECISION may also be specified to further describe the binary value. When a decimal value is converted to binary format, substring position and length, from 1 to 8 bytes, may be used to specify the substring of the target variable into which the binary source value is to be stored. For example, &target(,32,4), specifies that a 4-byte binary value is to be stored at position 32 of &target. When a binary value is converted to decimal format, adverbs SIGN, PUNCTUATE and JUSTIFY may be used to edit the decimal result.

- BITS converts source data that contains only the characters 0 and 1, to binary digits when TO BITS is specified, or converts the binary digits in the source data to the characters 0 and 1, when FROM BITS is specified. A binary digit is a 1-bit data item that occupies each bit of an 8-bit byte of data. When data is converted TO BITS, groups of 8 source characters are converted to binary digits and are stored in the target variable as a single 8-bit character of data. When data is converted FROM BITS, the 8 binary digits in each byte of source data are each converted to one of the 8-bit EBCDIC characters, 0 or 1, and are stored in the target variable as 8 characters of data.
- CAPITAL converts the alphabetic characters in the source data to upper case characters when TO CAPITAL is specified, or converts the alphabetic characters in the source data to lower case characters when FROM CAPITAL is specified. Non-alphabetic characters in the source data are copied to the target variable unchanged.
- DECIMAL states that the specified source data item contains a decimal value, and may be used to extract the decimal value from a virtual terminal screen field or from a character variable field. The extracted decimal value is subsequently used as though it were an actual decimal value.
- HEX converts source data that contains only the characters 0 through 9 and A through F, to hexadecimal digits when TO HEX is specified, or converts the hexadecimal digits in the source data to the characters 0 through 9 or A through F, when FROM HEX is specified. A hexadecimal digit is a 4-bit data item that occupies either the left-most or right-most 4 bits in an 8-bit byte of data. When data is converted to hexadecimal format, pairs of the source characters are converted to hexadecimal digits and are stored in the target variable as a single 8-bit character of data. When data is converted from hexadecimal format, each 4-bit hexadecimal digit of source data is converted to one of the 8-bit EBCDIC characters 0 through 9 or A through F, and is stored in the target variable as a single character of data.
- IMAGE edits the TRACKEDor virtual terminal screen field specified as the source data, and assigns the edited value to the target variable. On a real 3270 display, attributes, binary zeros and data contained in a non-display screen fields appear to be blanks. The editing performed by IMAGE converts 3270 attributes, binary zeros and data contained in non-display screen fields to blanks, and graphic escape characters to dashes. As a result, the image of a screen field is similar to the apparent value of a screen field when viewed on a real 3270 display. Because the image of a screen field does not contain attributes, binary zeros or non-display data, a script may compare the image of a screen field to an expected literal value without considering whether screen locations that appear to be blank are actually 3270 attributes, binary zeros or non-display data values.

- LENGTH assigns the length of the specified source data item as the value of the target variable, and target becomes a numeric variable. When a variable length record is read into a variable field, a SET statement may use LENGTH to determine the length of the variable field, and thus determine the length of the record.
- PACKED converts a decimal source value to packed format when TO PACKED is specified, or converts a packed source value to decimal format when FROM PACKED is specified. When PACKED is specified, the adjective, PRECISION may also be specified to further describe the packed value. When a decimal value is converted to packed format, substring position and length, from 1 to 16 bytes, may be used to specify the substring of the target variable into which the packed source value is to be stored. For example, &target(,20,9), specifies that a 9-byte packed value is to be stored at position 20 of &target. When a packed value is converted to decimal format, adverbs SIGN, PUNCTUATE and JUSTIFY may be used to edit the decimal result.
- PRECISION n may be specified when BINARY, PACKED or ZONED is specified, to state that a binary, packed or zoned decimal value, when expressed as a decimal value, has n decimal places. When PRECISION is specified and a binary, packed or zoned decimal value is converted to decimal, a decimal point is inserted in the decimal result in the proper position. When PRECISION is specified and a decimal value is converted to binary, packed or zoned decimal format, the decimal value is rounded to n decimal places by truncating excess decimal places or by adding zero digits as needed, after which the decimal point is removed and the resulting value is converted to a binary, packed or zoned decimal integer.
- SIGNED may be specified when BINARY is specified to state that a binary data item is signed. The high order bit (left-most bit), of a signed binary data item is used to represent the algebraic sign of the binary number. When the bit value is zero, the binary number is positive. When the bit value is one, the binary number is negative. When BINARY is specified but SIGNED is not, then the binary data item is unsigned. All bits in an unsigned binary data item are a part of the binary number, which is considered to be a positive number.
- STRING specifies that source consists of a list of one or more and that the character strings are to be concatenated to obtain the value to be assigned to the target variable. A SET statement that specifies no adjectives or adverbs and that specifies a list of data items as source, also concatenates the values of the listed data items, and thus STRING is not required to concatenate data. However, because STRING specifies that all specified source values are to be treated as character strings, STRING may be used to convert a numeric variable to a character variable.

- ZONED converts a decimal source value to zoned decimal format when TO ZONED is specified, or converts a zoned decimal source value to decimal format when FROM ZONED is specified. When ZONED is specified, the adjective, PRECISION may also be specified to further describe the zoned value. When a decimal value is converted to zoned format, substring position and length, from 1 to 30 bytes, may be used to specify the substring of the target variable into which the zoned decimal value is to be stored. For example, &target(,25,10), specifies that a 10-byte zoned decimal value is to be stored at position 25 of &target. When a zoned decimal value is converted to decimal format, adverbs SIGN, PUNCTUATE and JUSTIFY may be used to edit the decimal result.

Source is a required specification that represents the data that is to be copied, interrogated, converted, concatenated or otherwise manipulated to obtain the new target variable value. Source may be a literal, a decimal number, a screen field, a variable field, a substring of a variable field, an entry in a variable field array, may be a list of these elements, or may be the special symbol, asterisk, * . The special symbol, asterisk, *, specifies that the source value is the line number of the Script Language statement that is currently executing.

A single source data item is treated as either a decimal or character value. A single source data item is treated as a decimal value when it is an actual decimal number, when it is a numeric variable field, when the DECIMAL adjective is specified, when source data is converted TO BINARY, TO PACKED or TO ZONED, or when any of the adverbs, SIGN, PUNCTUATE or JUSTIFY, are used to edit a decimal value. In any other case, a single source data item is treated as a character value. When source is more than one data item, all source values are treated as character values and are concatenated to create the value to be assigned to the target variable.

adverb is one or more of the optional keywords, ONLY, SIGN, PUNCTUATE, or JUSTIFY. adverb may be specified only when TO or FROM is specified.

- ONLY may be specified when the target variable is an entry in a variable field array to specify that a value is to be assigned only to the specified array entry, and that if lower numbered array entries do not exist, that all non-existent lower numbered array entries are to be assigned an initial zero-length null value when they are created during SET statement processing. When a value is assigned to a specified entry in a variable field array and ONLY is not specified, and lower numbered array entries do not exist, then all non-existent lower numbered array entries are created and the value assigned to the specified array entry is propagated into all newly created lower numbered array entries.
- SIGN edits a decimal value by inserting an explicit leading sign, +or - , into the value assigned to the target variable.
- PUNCTUATE edits a decimal value by inserting commas (or periods if DECIMAL=COMMA was specified as an installation or run-time option), into the integer part of the value assigned to the target variable, to separate groups of 3 integer digits beginning at and to the left of the ones position.

- JUSTIFY edits a decimal value and will position the value in a specified substring of the target variable. When JUSTIFY is specified, the target variable field expression must specify a target substring position and length. For example, &target(,1,10), places the decimal value in a 10 character field that begins in position 1 of &target. JUSTIFY LEFT causes the value to be aligned in the left-most position of the target field. JUSTIFY RIGHT causes the value to be aligned in the right-most position of the target field. JUSTIFY CENTER left-justifies the sign and right-justifies the number in the target field.

Assign a Value to a Variable

The SET statement may be used to assign a value to or modify the value of a variable field, an entry in a variable field array, or a substring of a character variable field.

Examples

```
set &a-switch     TO 'yes'
set &b-entry(1)   to &a-switch only
set &b-entry(&c)  to <&d-row &e-col &f-len> only
set &g-field  to 'AB DE'set &g-field(,3,1) to 'C'
```

After execution of the SET statements in the previous examples, the script variable pool might contain the following data:

```
VARIABLE POOL DUMP
  a-switch     C'yes'         X'A885A2'
               Character variable; quotation marks delimit the
               literal and are not part of the variable
  b-entry(1)    C'yes'        X'A885A2'
  b-entry(5)    C'ABC12345'   X'C1C2C3F1 F2F3F4F5'
  c             N'5'
  d-row         N'12'
  e-col         N'52'
  f-len         N'8'
                The 5th occurrence in a variable array called
                "b-entry" was assigned a value obtained from an
                8 byte virtual terminal screen field at screen
                row 12, column 52
  g-field       C'ABCDE'      X'C1C2C3C4 C5'
                After setting "g-field" to "AB DE", a "C" was
                inserted in position 3 of "f-field", resulting
                in a field value of "ABCDE"
```

Propagate Data in a Variable Substring

The SET statement may be used to propagate a source value through a specified substring of a target variable field as many times as it will fit. Target field substring position and length control the extent of the propagation. If a substring length is not specified, the source value is propagated one time only, beginning at the specified substring position.

Examples

```
set &line(,1,24) to all '-'
set &hex-FF      to hex 'FF'
set &line(,17,4) to all &hex-FF
SET &PATTERN(,1,13) TO ALL "$-*-"
```

After execution of the SET statements in the previous examples, the script variable pool would contain the following data:

```
VARIABLE POOL DUMP:
  line    C'----------------'  X'60606060 60606060 60606060 60606060'
          C'....----'          X'FFFFFFFF 60606060'
 PATTERN  C'$-*-$-*-$-*-$'     X'5B605C60 5B605C60 5B605C60 5B'
```

Propagate Data in a Variable Array

The SET statement may be used to propagate values throughout a variable field array, or assign a value to a single entry in a variable field array.

Examples

```
drop &amount-due &counter &flag  /* delete existing arrays */
SET &AMOUNT-DUE(3) TO 1425.00 ONLY
set &counter(4)    to 0
set &counter(3)    to 3
SET &FLAG(2)       TO 'Y'
```

After executing of the previous statements, the CA Enterprise Tester variable pool would contain the following data:

```
VARIABLE POOL DUMP:
  AMOUNT-DUE(0)   C''  X''
  AMOUNT-DUE(1)   C''  X''
  AMOUNT-DUE(2)   C''  X''
  AMOUNT-DUE(3)   N'1425.00'
                  Occurrences 0, 1, and 2 of "&AMOUNT-DUE" are
                  zero-length null values in the variable pool
  counter(0)      N'0'
  counter(1)      N'0'
  counter(2)      N'0'
```

```
counter(3)      N'3'
counter(4)      N'0'
FLAG(0)      C'Y' X'E8'
FLAG(1)      C'Y' X'E8'
FLAG(2)      C'Y' X'E8'
```

CA Enterprise Tester variable memory does not exist until a value is assigned to it. When a value is assigned to an entry in an array, if the entry did not previously exist, the new entry and any lower-numbered entries that did not previously exist are created. By default, if any lower-numbered entries are created, the SET statement source value is propagated into all new entries. This feature allows an entire array of predictable size to be initialized to a desired value using a single SET statement. The ONLY keyword may be used to turn off propagation. When ONLY is specified and it is necessary to create new lower-numbered array entries, they are assigned zero-length null values. Once a variable field array entry is created, it exists until the entire array is deleted from the variable pool using the DROP statement.

Concatenate a List of Data Items

The SET statement may be used to construct a target field value by concatenating a list of data items. Data may be concatenated into a variable field, an entry in a variable field array or into a substring of a character variable field

Example

```
SET &ERR-MSG TO 'RECORD NO. ' &KEY ' CONTAINS INVALID '
                      &X '/' &Y '/' &Z(,3,2)
```

After execution of the SET statement in the previous example, the script variable pool might contain the following data:

```
VARIABLE POOL DUMP:
  ERR-MSG  C'RECORD NO. 9311 '  X'D9C5C3D6 D9C440D5 D64B40F9 F3F1F140'
           C'CONTAINS INVALID'  X'C3D6D5E3 C1C9D5E2 40C9D5E5 C1D3C9C4'
           C' AA/427/--'        X'40C1C161 F4F2F761 6060'
  KEY      N'9311'
  X        C'AA'                X'C1C1'
  Y        N'427'
  Z        C'14--972211'        X'F1F46060 F9F7F2F2 F1F1'
```

Initialize, Increment or Decrement a Variable

The SET statement may be used to assign a value to, increment, or decrement a numeric variable field.

Examples

```
set &total to 0
set &row to 23
do until &row = 5
   set &total up by <&row 55 10>
   set &row down by 2
enddo
```

After execution of the SET statements in the previous examples, the script variable pool would contain the following data:

```
VARIABLE POOL DUMP:
  row     N'5'
  total   N'1520'
```

Edit Decimal Data

Both the Script Language COMPUTE statement and the SET statement may specify decimal editing adverbs, SIGN, PUNCTUATE and JUSTIFY, that allow decimal results to be edited into neatly aligned report columns. SIGN forces an explicit sign to precede the decimal number. PUNCTUATE punctuates the integer part of a number into groups of 3 integer digits. JUSTIFY LEFT, CENTER, or RIGHT allows a number to be specifically positioned in a target variable field substring.

Examples

```
set &a-number        to 1234567.89
set &b-right(,1,15)  to &a-number sign punctuate justify right
set &c-center(,1,15) to &a-number sign punctuate justify center
set &d-left(,1,15)   to &a-number sign punctuate
```

After execution of the SET statements in the previous examples, the script variable pool would contain the following data:

```
VARIABLE POOL DUMP:
  a-number N'1234567.89'
  b-right  C'  +1,234,567.89'  X'40404EF1 6BF2F3F4 6BF5F6F7 4BF8F9'
  c-center C'+  1,234,567.89'  X'4E4040F1 6BF2F3F4 6BF5F6F7 4BF8F9'
  d-left   C'+1,234,567.89 '   X'4EF16BF2 F3F46BF5 F6F74BF8 F94040'
```

The editing capabilities of the SET statement and COMPUTE statement may be used to create formatted columner output.

For example:

```
  write outdd from 'ACCOUNT NO '
                   '  Binary Count'
                   ' Packed Amount'
                   '  Zoned Ratio '
Readloop
  read indd into &record endfile eof
  set &account   to &record(,1,10)
  set &col-1(,1,15) from signed binary &record(,25,9)
                        sign punctuate justify center
  compute &tot-1 = &tot-1 + &col-1
  set &col-2(,1,15) from packed precision 2 &record(,40,9)
                        sign punctuate justify center
  compute &tot-2 = &tot-2 + &col-2
  set &col-3(,1,15) from zoned precision 4 &record(,78,9)
                        sign punctuate justify center
  compute &tot-3 = &tot-3 + &col-3
  write outdd from &account ' ' &col-1 ' ' &col-2 ' ' &col-3
  goto readloop
eof
  compute &col-1(1,15) = &tot-1   sign punctuate justify center
  compute &col-2(1,15) = &tot-2 round 2 sign punctuate justify center
  compute &col-3(1,15) = &tot-3 round 4 sign punctuate justify center
  write outdd from 'TOTAL   ' &col-1 ' ' &col-2 ' ' &col-3
```

As an alternative to creating an output file, the WRITE statements could be replaced with LOG statements which would cause the formatted output to be written to the CA Enterprise Tester log data set.

Convert Binary Data to or from Decimal Format

The SET statement TO BINARY option allows a script to convert decimal numbers to binary format. The SET statement FROM BINARY option allows a script to convert binary numbers to decimal format. The SET keyword, SIGNED is used to identify signed binary numbers. While binary numbers are defined as integer values, the SET statement PRECISION keyword allows a script to specify an implied number of decimal places to be used when binary numbers are converted to or from decimal format.

Examples

```
set &a-pos-nbr to   123456
set &b-bin-nbr-1  to   binary &a-pos-nbr
set &c-bin-nbr-2(,1,4)  to   binary precision 2 &a-pos-nbr
set &d-dec-nbr-1  from binary &b-bin-nbr-1
set &e-dec-nbr-2  from binary precision 2 &b-bin-nbr-1
set &f-dec-nbr-3  from binary &c-bin-nbr-2
set &g-dec-nbr-4  from binary precision 2 &c-bin-nbr-2
```

```
set &h-neg-nbr to   -123456
set &i-bin-nbr-3  to   signed binary &h-neg-nbr
set &j-bin-nbr-4(,1,4)  to   signed binary precision 2 &h-neg-nbr
set &k-dec-nbr-5  from signed binary &i-bin-nbr-3
set &l-dec-nbr-6  from signed binary precision 2 &i-bin-nbr-3
set &m-dec-nbr-7  from signed binary &j-bin-nbr-4
set &n-dec-nbr-8  from signed binary precision 2 &j-bin-nbr-4
```

After execution of the SET statements in the previous example, the script variable pool would contain the following data:

```
VARIABLE POOL DUMP:
  a-pos-nbr   N'123456'
  b-bin-nbr-1 C'.S '    X'01E240'
  c-bin-nbr-2 C'.+/.'   X'00BC6100'
  d-dec-nbr-1 N'123456'
  e-dec-nbr-2 N'1234.56'
  f-dec-nbr-3 N'12345600'
  g-dec-nbr-4 N'123456.00'
  h-neg-nbr   N'-123456'
  i-bin-nbr-3 C'°.{'    X'FE1DC0'
  j-bin-nbr-4 C'.„°.'   X'FF439F00'
  k-dec-nbr-5 N'-123456'
  l-dec-nbr-6 N'-1234.56'
  m-dec-nbr-7 N'-12345600'
  n-dec-nbr-8 N'-123456.00'
```

If a target substring length is not specified when a decimal number is converted TO BINARY, then the target field length is determined to be the length of the smallest binary number that can contain the significant digits found in the source field. If the SIGNED keyword is specified, the high order bit in the binary result is used as the algebraic sign of the number, otherwise, all bits in the resulting binary number are a part of the number.

Convert Packed Data to or from Decimal Format

The SET statement TO PACKED option allows a script to convert decimal numbers to packed format. The SET statement FROM PACKED option allows a script to convert packed numbers to decimal format.

Examples

```
set &a-pos-nbr to   123456
set &b-pac-nbr-1  to   packed  &a-pos-nbr
set &c-pac-nbr-2(,1,10) to   packed  precision 2 &a-pos-nbr
set &d-neg-nbr to   -123456
set &e-pac-nbr-3  to   packed  &d-neg-nbr
set &f-pac-nbr-4(,1,10) to   packed  precision 2 &d-neg-nbr
set &g-dec-nbr-1  from packed &b-pac-nbr-1
```

```
set &h-dec-nbr-2  from packed precision 2 &b-pac-nbr-1
set &i-dec-nbr-3  from packed &c-pac-nbr-2
set &j-dec-nbr-4  from packed precision 2 &c-pac-nbr-2
set &k-dec-nbr-5  from packed &e-pac-nbr-3
set &l-dec-nbr-6  from packed precision 2 &e-pac-nbr-3
set &m-dec-nbr-7  from packed &f-pac-nbr-4
set &n-dec-nbr-8  from packed precision 2 &f-pac-nbr-4
```

After execution of the SET statements in the previous example, the script variable pool would contain the following data:

```
VARIABLE POOL DUMP:
  a-pos-nbr   N'123456'
  b-pac-nbr-1 C'.. ?'   X'0123456F'
  c-pac-nbr-2 C'....... -.'    X'00000000 00012345 600F'
  d-neg-nbr   N'-123456'
  e-pac-nbr-3 C'.. _'   X'0123456D'
  f-pac-nbr-4 C'....... -.'    X'00000000 00012345 600D'
  g-dec-nbr-1 N'123456'
  h-dec-nbr-2 N'1234.56'
  i-dec-nbr-3 N'12345600'
  j-dec-nbr-4 N'123456.00'
  k-dec-nbr-5 N'-123456'
  l-dec-nbr-6 N'-1234.56'
  m-dec-nbr-7 N'-12345600'
  n-dec-nbr-8 N'-123456.00'
```

If a target substring length is not specified when a decimal number is converted TO PACKED, then the target field length is determined to be the length of the smallest packed number that can contain the significant digits found in the source field.

Convert Zoned Decimal Data to or from Decimal Format

The Script Language SET statement TO ZONED option allows a script to convert decimal numbers to zoned decimal format. The SET statement FROM ZONED option allows a script to convert zoned decimal numbers to decimal format.

Examples

```
set &a-pos-nbr to   123456
set &b-chr-nbr-1(,1,10) to   string  &a-pos-nbr
set &c-dec-nbr-1  to   decimal &a-pos-nbr
set &d-zon-nbr-1  to   zoned &a-pos-nbr
set &e-zon-nbr-2(,1,10) to   zoned precision 2 &a-pos-nbr
set &f-neg-nbr to   -123456
set &g-chr-nbr-2(,1,10) to   string  &f-neg-nbr
set &h-dec-nbr-2  to   decimal &f-neg-nbr
```

```
set &i-zon-nbr-3  to   zoned &f-neg-nbr
set &j-zon-nbr-4(,1,10) to   zoned precision 2 &f-neg-nbr
set &k-dec-nbr-3  from zoned &d-zon-nbr-1
set &l-dec-nbr-4  from zoned precision 2 &d-zon-nbr-1
set &m-dec-nbr-5  from zoned &e-zon-nbr-2
set &n-dec-nbr-6  from zoned precision 2 &e-zon-nbr-2
set &o-dec-nbr-7  from zoned &i-zon-nbr-3
set &p-dec-nbr-8  from zoned precision 2 &i-zon-nbr-3
set &q-dec-nbr-9  from zoned &j-zon-nbr-4
set &r-dec-nbr-a  from zoned precision 2 &j-zon-nbr-4
```

After execution of the SET statements in the previous example, the script variable pool would contain the following data:

```
VARIABLE POOL DUMP:
  a-pos-nbr   N'123456'
  b-chr-nbr-1 C'123456 '   X'F1F2F3F4 F5F64040 4040'
  c-dec-nbr-1 N'123456 '
  d-zon-nbr-1 C'123456'    X'F1F2F3F4 F5F6
  e-zon-nbr-2 C'0012345600'    X'F0F0F1F2 F3F4F5F6 F0F0'
  f-neg-nbr   N'-123456'
  g-chr-nbr-2 C'-123456  '   X'60F1F2F3 F4F5F640 4040'
  h-dec-nbr-2 N'-123456'
  i-zon-nbr-3 C'12345O'    X'F1F2F3F4 F5D6'
  j-zon-nbr-4 C'001234560}'    X'F0F0F1F2 F3F4F5F6 F0D0'
  k-dec-nbr-3 N'123456'
  l-dec-nbr-4 N'1234.56'
  m-dec-nbr-5 N'12345600'
  n-dec-nbr-6 N'123456.00'
  o-dec-nbr-7 N'-123456'
  p-dec-nbr-8 N'-1234.56'
  q-dec-nbr-9 N'-12345600'
  r-dec-nbr-a N'-123456.00'
```

If a target substring length is not specified when a decimal number is converted TO ZONED, then the target field length is determined to be the length of the smallest zoned decimal number that can contain the significant digits found in the source field.

Convert Character Data to or from Decimal Format

The SET statement may be used to convert a numeric variable value to character format or to extract a decimal number from any data item and store it in a numeric variable field.

Examples

```
set &mm-dd-yy   to '06-01-94'
set &m  to decimal &mm-dd-yy(,1,2)
```

```
set &d  to decimal &mm-dd-yy(,4,2)
set &y  to decimal &mm-dd-yy(,7,2)
set &m-d-y   to string &m '/' &d '/' &y
set &screen-amt to decimal <12 40 10>
set &chr-string to string  &screen-amt
```

After execution of the SET statements in the previous examples, the script variable pool might contain the following data:

```
VARIABLE POOL DUMP:
  mm-dd-yy C'06-01-94' X'F0F660F0 F160F9F4'
  m  N'6'
  d  N'1'
 y  N'94'
  m-d-y C'6/1/94'  X'F661F161 F9F4'
  screen-amt  N'1123.99'
  chr-string  C'1123.99' X'F1F1F2F3 4BF9F9'
```

Convert Data to or from Hexadecimal Format

The SET statement FROM HEX option may be used to display a record or other data that contains non-printable characters, by converting 4-bit hexadecimal digits in the source data to the characters 0 through 9 and A through F. The SET statement TO HEX option may be used to convert pairs of the characters 0 through 9 and A through F, to hexadecimal digits which are stored as a single byte of data.

Examples

```
read indd into &a-in-record
set &b-display from hex &a-in-record(,3,2)
set &c-display from hex &a-in-record(,20,2)
log 'invalid data area 1 was: ' &b-display
log 'invalid data area 2 was: ' &c-display
pause /*input data via Interactive System EXECUTE command/*
set &f-out-record to &a-in-record
set &f-out-record(,3,2) to   hex &d-correction-1
set &f-out-record(,20,2) to   hex &e-correction-2
write outdd from &f-out-record
set &hex-ff to hex 'ff'
```

After execution of the SET statements in the previous example, the script variable pool might contain the following data:

```
VARIABLE POOL DUMP:
  a-in-record C'ab..efghijklimop' X'81820304 85868788 89919293 89949697'
              C'qrs..vwxyz  '     X'9899A203 04A5A6A7 A8A94040'
  b-display   C'0304'   X'F0F3F0F4'
  c-display   C'0304'   X'F0F3F0F4'
```

```
  d-correction-1  C'8384'   X'F8F3F8F4'
  e-correction-2  C'A3A4'   X'C1F3C1F4'
  f-out-record C'abcdefghijklimop' X'81828384 85868788 89919293 89949697'
               C'qrstuvwxyz  '  X'9899A2A3 A4A5A6A7 A8A94040'
  hex-ff   C'.'   X'FF'
```

Manipulate Bit Patterns

The SET statement TO BITS option may be used to convert source data containing a string of character zeros and ones to bytes whose bit values match the source zeros and ones. The SET statement FROM BITS option may be used to convert one or more bytes of source data to a string of character zeros and ones that correspond to the bit values contained in the source bytes.

Examples

```
read indd into &a-in-record
set &b-display-flag from bits &a-in-record(,25,1)
set &c-change-flag  to &b-display-flag
set &d-date  to &a-in-record(,14,6)
if &d-date gt 930000
then
  set &c-change-flag(,1,1) to '1'
  set &c-change-flag(,3,1) to '1'
  set &c-change-flag(,5,1) to '1'
  set &c-change-flag(,7,1) to '1'
endif
log 'flags were: ' &b-display-flag
log 'flags are now: ' &c-change-flag
set &e-out-record to   &a-in-record
set &e-out-record(,25,1) to bits &c-change-flag
write outdd from &e-out-record
```

After execution of the SET statements in the previous example, the script variable pool would contain the following data:

```
VARIABLE POOL DUMP:
 a-in-record   C'SMITH, JAMES 931' X'E2D4C9E3 C86B40D1 C1D4C5E2 40F9F3F1'
               C'205 12 x¡' X'F2F0F540 F1F240A7 55'
 b-display-flag  C'01010101'  X'F0F1F0F1 F0F1F0F1'
 c-change-flag   C'11111111'  X'F1F1F1F1 F1F1F1F1'
 d-date C'931205' X'F9F3F1F2 F0F5'
 e-out-record  C'SMITH, JAMES 931' X'E2D4C9E3 C86B40D1 C1D4C5E2 40F9F3F1'
               C'205 12 x.' X'F2F0F540 F1F240A7 FF'
```

Change the Case of Alphabetic Data

The SET statement TO CAPITAL option may be used to capitalize alphabetic characters contained in a source field. The SET statement FROM CAPITAL option may be used to convert capital alphabetic characters contained in a source field to lowercase characters.

Examples

```
set &b-all-lc from capital &a-mixed-case
set &c-all-uc to   capital &a-mixed-case
```

After execution of the SET statements in the previous example, the script variable pool would contain the following data:

```
VARIABLE POOL DUMP:
a-mixed-case   C'aBcDeFgHiJkLmNoP' X'81C283C4 85C687C8 89D192D3 94D596D7'
               C'qRsTuVwXyZ012345' X'98D9A2E3 A4E5A6E7 A8E9F0F1 F2F3F4F5'
               C'6789'             X'F6F7F8F9'
b-all-lc       C'abcdefghijklmnop' X'81828384 85868788 89919293 94959697'
               C'qrstuvwxyz012345' X'9899A2A3 A4A5A6A7 A8A9F0F1 F2F3F4F5'
               C'6789'             X'F6F7F8F9'
c-all-uc       C'ABCDEFGHIJKLMNOP' X'C1C2C3C4 C5C6C7C8 C9D1D2D3 D4D5D6D7'
               C'QRSTUVWXYZ012345' X'D8D9E2E3 E4E5E6E7 E8E9F0F1 F2F3F4F5'
               C'6789'             X'F6F7F8F9'
```

Determine Length of a Field of Data

The SET statement may be used to determine the length of a variable field or a virtual terminal screen field.

Examples

```
  set &record-len  TO   LENGTH &record
  set &screen-len  to   length <7,24>
```

After execution of the SET statements in the previous examples, the script variable pool would contain the following data:

```
VARIABLE POOL DUMP:
  record      C'This record is ' X'E38889A2 40998583 96998440 89A24040'
              C'30 chars long '  X'F3F24083 888199A2 40939695 8740'
  record-len  N'30'
  screen-len  N'57'
```

Screen field length is measured from the specified row and column, to the next screen location that is an attribute or to the end of the screen

Edit a Virtual Terminal Screen

Variables which contain data retrieved from a virtual terminal screen may be created using the Script Language SET statement. The SET statement TO IMAGE option may be used to create a data element that contains the visible contents of a virtual terminal screen field or a TRACKED screen field. The editing performed by IMAGE converts 3270 attributes, binary zeros and data contained in non-display screen fields to blanks, and graphic escape characters to dashes.

Examples

```
set &a-title  to <1,1,80>
set &b-image-title to image <1,1,80>
```

After execution of the SET statements in the previous example, the script variable pool might contain the following data:

```
VARIABLE POOL DUMP:
 a-title       C'................' X'11000000 00000000 00000000 00000000'
               C'Accounts.Payable' X'C1838396 A495A3A2 00D781A8 81829385'
               C'.Vendor.Master.F' X'00E58595 84969900 D481A2A3 859900C6'
               C'ile.Maintenance.' X'89938500 D4818995 A3859581 95838500'
               C'.........93262..' X'00000000 00000000 17F9F3F2 F6F20000'
b-image-title  C'                ' X'40404040 40404040 40404040 40404040'
               C'Accounts Payable' X'C1838396 A495A3A2 40D781A8 81829385'
               C' Vendor Master F' X'40E58595 84969940 D481A2A3 859940C6'
               C'ile Maintenance ' X'89938540 D4818995 A3859581 95838540'
               C'                ' X'40404040 40404040 40404040 40404040'
```

Determine Attributes of a Screen Location

The SET statement ATTRIBUTES adjective may be used to generate a list of keywords that describes all of the 3270 attributes of a virtual terminal screen location or a TRACKED screen location. The ATTRIBUTES adjective must be followed by a screen field expression, or by the TRACKED adjective to refer to a screen location in a screen image that was read from a CA Enterprise Tester trace data set using the TRACK statement.

Example

```
set &attr     to attributes <4,7>
```

After execution of the SET statement in the previous example, the script variable pool might contain the following data:

```
VARIABLE POOL DUMP:
  ATTR  C'FIELD PROTECTED '  X'C6C9C5D3 C440D7D9 D6E3C5C3 E3C5C440'
        C'NORMAL BLUE'       X'D5D6D9D4 C1D340C2 D3E4C5'
```

A list of all possible attribute names is contained in 3270 Attribute Values.

After obtaining a list of the attributes at a screen location, a script may use the CONTAINS condition in an IF or DO statement to examine a particular property of a screen location.

For example:

```
SET &attr to ATTRIBUTES <1 2>
IF &attr CONTAINS word 'modifiable'
then
   type <1 2> 'x'
else
   log 'row 1, column 2 is not modifiable'
endif
```

Scripts that use the ATTRIBUTES adjective to access screen attributes are independent of differences in actual attribute values that are stored in either basic or extended data stream virtual terminal buffers. The ATTRIBUTES adjective is the only way for a script to access attributes that are stored in a virtual terminal extended attribute buffer.

Save the Executing Line Number of a Script

The SET statement may be used to capture the script statement line number of the SET statement being executed.

Example

```
set &line-no to *
log &line-no
```

After execution of the SET statement in the previous example, the variable, &line-no, would contain the script statement line number of the Script Language statement, set &line-no to *.

THEN

The THEN verb is used in an IF statement to identify a routine that is to be executed when specified conditions are true. Refer to IF for details.

THINK

The THINK statement is used to control the speed at which a script enters transactions on a virtual terminal session. A THINK statement may specify an amount of time, called think time, that script execution should be delayed prior to sending each transaction entered by a subsequent Script Language TYPE statement. Think time delays simulate the session delays that occur on real 3270 sessions when a person gets a response from one transaction, then thinks about it or performs some other activity before entering the next transaction.

The THINK statement may also be used to control the workload generated by one or more CA Enterprise Tester scripts. The THINK statement RATE option causes think times to be automatically calculated between transactions, that result in entering a specified number of transactions per second or a specified number of transactions in a given period of time. This statement has the following format:

```
THINK hhmmssth |
   ( RATE transactions
      ( PER pertime )
      ( SESSIONS sessions )
      ( RTIME resptime )
      ( SAMPLE samples  ) )
```

hhmmssth

A think time interval, expressed in hours (hh), minutes (mm), seconds (ss) and tenths and hundredths of seconds (th). A think time interval may be specified using a character variable field that contains the interval in the format, hhmmssth. One and only one specification of hhmmssthand RATE, is required.

RATE transactions

Specifies a number of transactions to be entered per second or the number of transactions to be entered in a given period of time when PERpertimeis specified. Transactions may be an integer greater than zero or may be a variable field that contains the number of transactions to be entered in the default or specified period of time.

PER pertime

Optional and may be used to specify the time interval in which the specified number of transactions are to be entered. pertime may be expressed in the form, hhmmssth or may be a character variable that contains the time interval in the form, hhmmssth. When RATE is specified, but PER is not specified, pertime defaults to 100 (one second).

SESSIONS

Optional, and may be used to specify the total number of concurrently executing scripts or sessions that are to enter the transactions specified by the RATE and PER options. Sessions may be an integer greater than zero or may be a variable field that contains the number of sessions that are to enter transactions. When RATE is specified, but SESSIONS is not specified, sessions defaults to 1. When sessions is greater than 1, the script will attempt to enter transactions at the following rate per second:

```
(1 / sessions) * (transactions / pertime)
```

RTIME resptime

Optional, and may be used to estimate the response time expected on a session. During a session that is paced using THINK RATE, actual response time is monitored and the actual response time is used in calculating think times needed to enter transactions at a desired rate. The RTIME option provides a way to specify an initial estimate of response time to improve the calculation of think times prior to having measured any actual response times. resptime may be expressed in the form, hhmmssth or may be a character variable field that contains the response time estimate in the form, hhmmssth. When RATE is specified, but RTIME is not specified, resptimedefaults to the average response time on a session, if a session is active when the THINK statement is executed, otherwise resptime defaults to 200 (2 seconds).

SAMPLE samples

Optional, and may be used to specify the number of recent transaction response times to be considered when recalculating average response time. While a script is executing, that is being paced by a THINK RATE statement, exponential smoothing is used to recalculate the average response time being experienced on a session. Prior to entering each transaction, the following calculation is performed to smooth the most recent actual response time (actual-rtime), into the new average response time estimate:

```
new-average = (((samples -1) * old-average) + actual-rtime) / samples
```

Subsequently, think time is calculated as the time interval in which a transaction is to be entered, based on the specified rate, minus the new average response time. When samples is small, average response time, and thus the calculated think time, fluctuates significantly for each transaction. When samples is large, average response time converges to the average for the entire session and tends to result in uniform think times. Samples may be an integer from 1 to 1000 or may be a variable field that contains the number of recent transaction response times to be considered. When RATE is specified, but SAMPLE is not specified, samples defaults to 10.

Example

```
/* hhmmssth                                              */
THINK 10000        /* one minute                         */
THINK  3000        /* thirty seconds                     */
THINK 200          /* two seconds                        */
THINK 50            /* half of a second                  */
THINK RATE 500      /* pace script to enter              */
      PER 1000000   /* 500 transactions/hour             */
THINK RATE 25       /* pace script to enter 1/100 of total */
      SESSIONS 100  /* workload of 25 transactions/second  */
```

The think time delays activated by executing a THINK statement remain in effect until changed by executing another THINK statement. Think time delays may be inactivated by executing a THINK statement that specifies a think time of 0 (zero). Outbound session traffic may continue to flow to the virtual terminal during think time intervals.

When a script is played under the control of the CA Enterprise Tester Interactive System, other session delays occur when the WAIT command is used to view a session in movie mode or when the STEP command is used to step through script driven session activity one screen at a time. An option on the Interactive System Options Panel controls whether or not think time delays are to occur when script progress is also under the control of the STEP or WAIT command. The CA Enterprise Tester Options Panel is described in Options and Controls.

Controlling Multiple Session Tests illustrates how THINK or THINK RATE statements may be used in conjunction with the Interactive System EXECUTE command to control the speed of scripts or the total workload generated by scripts used in a multiple session test of an online system.

-- Heading id 'ussthin' unknown -- documents a sample CA Enterprise Tester script that selects random think times that may be used to pace sessions in ways other than those directly supported by the THINK statement.

CA Enterprise Tester also provides a DELAY statement which may represent actual user think time in a recorded script. The Record Facility may automatically capture DELAY intervals which were incurred during a recorded session. A DELAYPERCENT factor may be specified when a script is played to expand or compress the recorded delay intervals.

DELAY documents the statement processing, and you may also refer to Recording Session Scripts and Playing Session Scripts for more information on this topic.

TIME

The TIME statement provides the current time of day to a script. This statement has the following format:

Execution of a TIME statement sets character variable fields to the current time elements:

- PTEHR will be set to a value from 00 through 23.
- PTEMIN will be set to a value from 00 through 59.
- PTESEC will be set to a value from 00 through 59.
- PTETIME will be set to a 20 character string of the form, yyyymmddhhmmssthmiju, where the format characters represent: yyyy, year; mm, month; dd, day; hh, hour; mm, minutes; ss, seconds; th, tenths and hundredths of seconds; m, millisecond; i, ten-thousandths of seconds; j, hundred-thousandths of seconds; and u, micro-second.

TIME variable fields may not be modified by a script, but may otherwise be used wherever other character variable fields may be used.

The TIME variable fields are automatically set at the beginning of script execution and are updated only when the TIME statement is executed.

Generating Random Numbers

Using low order digits of the PTETIME variable, the COMPUTE statement can be used to generate a random number in a desired range.

For example:

```
time
compute &random = &ptetime(,17) / remainder &range + &base
```

computes a random number between &base and (&base + &range), using the low order 4 digits of PTETIME, which produces all possible 4 digit decimal integers once every 1/100 of a second.

TIMEOUT

The TIMEOUT statement is used to specify the maximum amount of time that CA Enterprise Tester should wait for the completion of an in-progress session event. The time limit is activated when a HANDLE TIMEOUT ON or HANDLE TIMEOUT AT statement has been executed to activate session time out processing. This statement has the following format:

```
TIMEOUT hhmmssth
    ( S522 OFF | ON )
```

hhmmssth

A time interval expressed in hours (hh), minutes (mm), seconds (ss) and tenths and hundredths of seconds (th). A TIMEOUT interval may be expressed using a character variable field containing data in the format, hhmmssth.

S522

An optional parameter which specifies that installation timeout abends should be controlled. OFF disables S522 abends; ON enables S522 abends.

Example

```
TIMEOUT 1000   /* ten seconds*/
HANDLE TIMEOUT ON
timeout s522 off   /* disable timeout abends */
```

When a HANDLE TIMEOUTON or AT statement has been executed, but a TIMEOUT statement has not been executed, the time out interval defaults to zero. Because no event can complete in zero time, a TIMEOUT statement that specifies a non-zero interval should be executed before executing a HANDLE statement to enable time out processing.

When time out processing is used, a session will not be started if a subsequent LOGON statement does not complete within the specified TIMEOUT interval.

When time out processing is used, execution of a TYPE or INVITE statement will complete if any single session event does not complete within the TIMEOUT interval. For example, when a TYPE statement is used to enter a transaction, the statement normally completes when the online system response has been received. When time out processing is used, if the online system response has not been received within the TIMEOUT interval, the TYPE statement will complete after waiting the specified amount of time for the response. A session is timed out when TYPE or INVITE statement completes prematurely as a result of time out processing.

When a session is timed out, an INVITE statement may be used to resume the wait for the completion of the timed out event. While timed out, a session may be in any of several states not normally seen by a script. The value of the variable, PTESTATE, identifies the state of the session. CA Enterprise Tester Logical Unit States documents all of the possible session states and the possible values of PTESTATE. Another variable field, PTEKB, identifies the state of the virtual terminal keyboard. While a session

is timed out, the keyboard will be in either a LOCKED or INHIBITED state. When the keyboard is LOCKED, the TYPE statement may not be used. If the keyboard is in the INHIBITED state, the TYPE statement may by used only to send an Attention Key signal to the online system.

For example:

```
TYPE ATTN
```

Scripts that invoke very long running transactions may incur a system generated S522 abend, sometimes referred to as a timeout, if the script waits for a transaction response longer than an installation controlled wait time limit. The TIMEOUT statement keyword, S522, may be used to disable S522 abends while a script is waiting on a transaction(s) that is known to require long processing times. TIMEOUT S522 OFF disables S522 abends; TIMEOUT S522 ON enables S522 abends.

TRACE

If a CA Enterprise Tester trace is started using the Interactive System TRACE command, session screens images are subsequently written to a trace data set when messages are sent or received on the traced session. Session screen images may also be traced from a CA Enterprise Tester script executed as a batch job.

When a PTETRACE DD statement is present in CA Enterprise Tester Batch System JCL, a session trace will automatically be initiated when the script is loaded for execution, will optionally be ended when script execution is ended, and will be stored in the data set specified by the PTETRACE DD statement.

The TRACE statement allows a script to selectively trace session interactions and to control the destination, format and contents of the trace. This statement has the following format:

```
TRACE ON | OFF
   ( TO ddname ( PASS ) )
   ( FORMAT | IMAGE | UPPERCASE )
   ( DELAYS | DELAYSOFF )
   ( LOGONS | LOGONSOFF )
   ( CHAINS | FREEKB )
```

```
( PACK | NOPACK )
( caption )
```

ON

Specifies subsequent session activity is to be traced;

OFF

Specifies subsequent session activity should not be traced;

TO ddname

Specifies the name of the data definition, (DD), to which subsequent screen images are to be written, opens the data set, and initiates the trace function;

PASS

Specifies that the trace is to remain active after a script played via the Interactive System ends.

FORMAT

Optional and specifies that traced screen images are to contain all screen data; a FORMAT trace is required as input to session analyzers.

IMAGE

Optional and specifies that traced screen images are to be edited to convert attributes and other non-display data to blanks, and graphic escape characters to dashes, producing screen images similar to screen images visible to users of the system.

UPPERCASE

Optional and specifies that traced screen images are to be edited and converted to all upper case characters;

DELAYS

Specifies that user think time is to be captured with DELAY statements in the trace data set; DELAYSOFF specifies that DELAY statements should not be created.

LOGONS

Specifies that logon statements should be captured with LOGON statements in the trace data set; LOGONSOFF specifies that LOGON statements should not be created.

CHAINS

Specifies that all outbound screen images should be traced; FREEKB specifies that only outbound screens that free the keyboard should be traced.

PACK

Optional and specifies that traced screen images are to be written using the ISPF packed data format; NOPACK specifies that traced screen images are to be written as unpacked, fixed-length records;

Caption

An optional list of data items, which may include variable fields, screen fields, literals and numbers whose values are to be concatenated to construct a screen caption, up to 40 characters long, that is to be included on subsequently traced screens.

Default values for all optional parameters are determined by system installation specifications.

Examples

```
TRACE TO DDTRAC1
TRACE FORMAT
```

TRACE statements that are executed when a trace data set is not available are tolerated and perform no operation.

A script may selectively control session trace activity.

- By default, if a PTETRACE DD statement is included in batch JCL, session screen images will be traced.
- A TRACE OFF statement may be executed to suppress tracing of subsequent session interactions. Turning a trace off does not close the trace data set, but rather simply suppresses the capture of screen images until the trace is turned back on.
- When a TRACE ON statement is executed, subsequent session interactions will be traced.
- An ENDTRACE statement ends the trace and closes the data set.
- A single trace data set may be active on a port. If a TRACE TO ddname has been executed, or a trace is active because of a PTETRACE DD statement, a subsequent TRACE TO ddname statement will end the active trace before starting the new one.
- PASS may be specified on a TRACE TOstatement to specify that the trace is to remain active after a script played via the Interactive System ends. When a script is played interactively, it may start a session, recording and/or trace, and end, leaving the started functions active to capture subsequent interactive session activity entered manually on the session started by the script. This technique may simplify session capture activity by eliminating the need to manually start a session, recording and/or trace.

The TRACE statement may be used to specify the content of traced screen images. The content of traced screen images may also be specified using a command when a trace is started using the CA Enterprise Tester Interactive System TRACE command or Trace Panel, or may be specified via the JCL PARM field keywords, FTRACE, ITRACE, and UTRACE, when a script is executed as a batch job.

- TRACE FORMAT specifies that traced screen images are to contain all screen data, including attributes, dark data and non-display data. When an extended data stream virtual terminal is used for a session, then a CA Enterprise Tester Format Trace also contains the extended attribute buffer for each traced screen.
- TRACE IMAGE specifies that traced screens are to be edited to convert attributes, dark data and other non-display characters to blanks, and graphic escape characters to dashes. An Image Trace looks like what a person sees on a screen and produces screen images that are suitable for inclusion in documentation of online systems. IMAGE is the default for a trace produced by a batch job.
- TRACE UPPERCASE specifies that Image Trace editing is to be performed and additionally that lowercase alphabetic characters in the traced screen and in trace data set headings and screen borders are to be converted to uppercase characters.
- The TRACE statement may be used to specify the type of outbound session screen images which should be included in the trace, and may be one of the keywords, CHAINS or FREEKB. The type of outbound images to include in the trace may also be specified at the Interactive System Trace Panel when a trace is started or may be specified using the JCL PARM field keywords, TCHAINSor TFREEKB, when a script is executed as a batch job in which the online session is to be traced. If an option is not specified, a default, specified during CA Enterprise Tester installation, is used.
- TRACE CHAINS specifies that all outbound session screen images are to be included in the trace data set. The result is that multiple outbound session images may be included for each inbound session application request. An example of an online application which generates multiple outbound screen images is the numerous outbound images which are typically presented during a TSO/ISPF/PDF sign on procedure.
- TRACE FREEKB specifies that only outbound session screens which free the terminal keyboard are to be included in the trace data set. This option may reduce the number of outbound screen images included in the trace data set.

- The TRACE statement may be used to specify options which control whether or not specific PSL statements should be included in the trace data set, and may be one of the keywords, LOGON or LOGONSOFF, or DELAYS or DELAYSOFF. Options for including PSL statements in the trace may also be specified via the Interactive System Trace Panel when a trace is started or may be specified using the JCL PARM field keywords, TLOGON or TLOGONSOFF; or TDELAY or TDELAYSOFF, when a script is executed as a batch job, and the session is to be traced. If an option is not specified, a default, specified during CA Enterprise Tester installation, is used.
- TRACE LOGONS specifies that the trace is to include PSL LOGON statements and parameters which capture all data necessary to reproduce the session logon procedure.
- TRACE LOGONSOFF specifies that PSL LOGON statements should not be included in the trace.
- TRACE DELAYS specifies that the trace is to include PSL DELAY statements which capture the actual times between transactions that occurred when the live session was captured. By default, traces that contain generated DELAY statements which are used as input to session analyzer scripts, will execute at the same speed as the captured session. (Or, more precisely, a very close approximation of the same speed, assuming all system variables are similar to those at capture time). However, a DELAY PERCENT may be specified that increases or decreases each DELAY interval by the specified percentage.
- TRACE DELAYSOFF specifies that DELAY statements should not be included in the trace. For more information on the DELAY statement, refer to DELAY . For more information on managing workloads delivered to a system please refer to sections which deal with multiple session tests.
- The TRACE statement may be used to specify the format of the data written to a trace data set and may be one of the keywords, PACK or NOPACK. The format of trace data may also be specified via the Interactive System Trace Panel when a trace is started or may be specified using the JCL PARM field keywords, PTRACE or NOPTRACE, when a script is executed as a batch job. If a format is not specified, a default, specified during CA Enterprise Tester installation, is used.
- TRACE PACK specifies that traced screen images are to be written to the trace data set using the ISPF packed data format. The packed data format significantly reduces the disk space required for a trace. Packed trace data sets appear to be ordinary data when accessed using ISPF/PDF EDIT or BROWSE or when processed using the CA Enterprise Tester Script Language TRACK statement, which is used in the distributed CA Enterprise Tester session analyzers. Packed traces may be unpacked using ISPF/PDF EDIT, by saving an edited trace after entering the PDF command, PACK OFF.
- TRACE NOPACK specifies that traced screens are to be written as normal unpacked fixed length records. Traces that are not packed may be printed using standard print utilities, including the TSO PRINTOFF command.

The TRACE statement may be used to construct a caption to be included on subsequently traced screens.

```
TRACE 'This is caption number ' &number
```

Additional TRACE statements may be executed to change the caption. The caption may be turned off by setting the caption to a null value.

For example:

```
TRACE ''
```

TRACK

The TRACK statement reads a screen image from a previously created CA Enterprise Tester Trace data set. A script may use the TRACKED adjective and standard screen field syntax to refer to screen fields in a screen image that was read by a TRACK statement. The TRACK statement also puts information about the screen image in CA Enterprise Tester defined variable fields. This statement has the following format:

```
TRACK ddname
   ( screen-type
     PHYSICAL | LOGICAL | TRANSACTION )
   ( record-type
     STATEMENT | LINE )
   ( terminal-type
     BASICDS | EXTDS )
   ( ENDFILE routine )
   ( ERROR routine | * )
```

ddname

The name of a data definition, (DD), statement that identifies a previously created CA Enterprise Tester trace data set. ddname may be the actual DD name or may be a variable field that contains the DD name;

screen-type

Optional, and may be one of the keywords PHYSICAL, LOGICAL or TRANSACTION, to specify the type of screen to be read.

- PHYSICAL, the default, specifies that the next sequential screen image is to be read.
- LOGICAL, specifies that only inbound screens or outbound screens that free the terminal keyboard are to be read.
- TRANSACTION, specifies that only inbound screens and outbound screens that precede an inbound screen are to be read.

An inbound screen is a screen that was sent from a virtual terminal to an online system. An outbound screen is a screen that was sent by an online system to a virtual terminal.

terminal-type

Optional, and may be one of the keywords BASICDS or EXTDS, to refer to whether or not the extended 3270 data stream is supported, and to the number of virtual terminal buffers required to store data for the screen image that is read.

BASICDS terminal type

Specifies that a single virtual terminal screen buffer is captured in the trace data set.

EXTDS terminal type

Specifies that the extended 3270 data stream is supported, and that two virtual terminal buffers are captured, the screen buffer and the extended attribute buffer.

record-type

Optional, and may be one of the keywords STATEMENT or LINE, to track any data in the trace data set which is not part of a virtual terminal screen image, nor part of trace page header records, but may be CA Enterprise Tester Script Language statements or comments.

STATEMENT

Specifies that if a PSL statement, for example a LOGON or DELAY statement, is the next data found in the dataset being tracked, then the PSL statement is to be returned to the script instead of a subsequent session screen image. When a TRACK statement specifies STATEMENT, only specified screens and complete PSL statements are returned to a script, and comments and screens that may be commented out are skipped.

LINE

Specifies that trace dataset logical records, or lines of data other than traced titles and screen data that are found in a format trace data set are to be returned to the script. When a TRACK statement specifies LINE, comment lines, including lines of a commented out screen image are returned to a script as lines.

When a TRACK statement specifies neither LINE nor STATEMENT, only screen images that are not commented out are returned to the script, and all comments, PSL statements or other non-screen trace data is skipped.

CA Enterprise Tester provides automatic unusual condition processing when an optional condition keyword is specified. Each condition keyword is followed by the name of a script routine that is to receive control if the condition is encountered.

A condition handling routine may be expressed as a statement label or number, as a variable field that contains a statement label or number, or may be the special symbol, asterisk, *, to specify that execution is to be resumed at the next script statement. Condition keywords supported by the TRACK statement are ENDFILE and ERROR.

ENDFILE routine

Identifies a script routine that is to receive control when an end-of-file condition is encountered during the execution of a TRACK statement. If ENDFILE is not specified on a TRACK statement, an ENDFILE routine specified on an associated OPEN statement will be used. If ENDFILE is not specified and the condition occurs, the end-of-file condition is considered to be an ERROR.

ERROR routine

Identifies a script routine that is to receive control when an error or other unhandled condition is encountered during the execution of a TRACK statement. If ERROR is not specified on a TRACK statement, an ERROR routine specified on the associated OPEN statement will be used.

Example

```
TRACK TRACE TRANSACTION ENDFILE EOF
```

The TRACK statement may be used to read a trace data set that was created using the Interactive System TRACE command and Trace Panel. The TRACK statement can not be used to read a trace data set that was captured using the IMAGE or UPPERCASE trace format. The trace data set must have data set characteristics: RECFM=FBA, and LRECL=135 or greater. After starting a trace, screen images that reflect session activity that occurs on the Interactive System port on which the trace is active, are captured in the trace data set.

The TRACK statement may also be used to process a trace data set that is written to a PTETRACE DD statement, when a script is executed using a batch job, if the FTRACE keyword was specified in the JCL PARM field or a Script Language TRACE FORMAT statement is executed prior to starting a session, and if the PTETRACEdata set has data set characteristics: RECFM=FBA, and LRECL=135 or greater.

The DD name specified on a TRACK statement must be opened by a script using the Script Language OPEN statement before the TRACK statement is executed.

The TRACK statement reads a trace data set sequentially and returns the next screen image or a screen image described by the LOGICAL or TRANSACTION keyword. The image is stored in one or two virtual terminal buffers, depending on the number of buffers captured, or is converted to the desired form, if needed, when the BASICDS or EXTDS keyword is specified. When a TRACK statement specifies BASICDS, a screen image is made available to a script in a single virtual terminal buffer, and attributes stored in the basic screen buffer are converted to coded basic field attributes if needed. When a TRACK statement specifies EXTDS, a screen image is made available to a script using a screen buffer and an extended attribute buffer. If needed, coded basic data stream attributes are converted to the actual attribute values used with extended data stream virtual terminals. You may refer to the topic, 3270 Attribute Values for a description of CA Enterprise Tester virtual terminal buffers and attribute values.

A script may use the TRACKED adjective and standard screen field syntax to refer to screen fields contained in the screen read by the TRACK statement.

For example:

```
if tracked <1 2 *> = 'CESN'
   go to maskcesn
type <&row 1> tracked <&row 1 &ptetcols>
set &line to image tracked <&row 1 80>
set &attr to attributes tracked <&row &col 1>
```

Non-display Fields and Security

When screen images that contain password tokens or codes are tracked, secured data is managed in memory by the active password security manager. When Token Security is active, non-display virtual terminal fields are non-display even to the script and both token and code data is maintained in memory in token form. When Code Security is active, data is decoded or decrypted using the user ID that owns the address space used to TRACK the data and is present in the non-display field. When either password security manager is used, if secure tracked data is typed into a non-display field of a live session screen, the correct password is sent when it used by the owner of the data even if it is not visible to the script.

Screen Image Sequence Numbers

Traces produced by CA Enterprise Tester Release 3.0 and subsequent releases contain a screen sequence number in the screen title. The TRACK statement returns the sequence number in variable field PTETSEQ, in the form #nnnnn, as it appears in the screen title. When a trace produced by an older release of CA Enterprise Tester is processed, TRACK sets PTETSEQ to the value, #00000.

Executable Code in a Trace Data Set

The TRACK statement keyword, STATEMENT, specifies that if a PSL statement, for example a LOGON or DELAY statement is the next data found in the dataset being tracked, the PSL statement is to be returned to the script instead of a subsequent session screen.

When STATEMENT is specified and a PSL statement is found, the entire statement is returned in variable field, PTETLINE and the TRACK statement ends with PTERESP set to the value, LINE. A statement returned in PTETLINE is left-justified, with the PSL verb beginning in position 1 of PTETLINE, and all insignificant blanks between statement keywords and data items, and any /* comments */ coded within the PSL statement are removed.

A statement returned by TRACK may be executed using the PSL EXECUTE statement.

For example:

```
EXECUTE &PTETLINE
```

Reading Logical Records in a Trace Data Set

The TRACK statement keyword, LINE, specifies that trace dataset logical records, or lines of data other than traced titles and screen data that are found in a format trace data set are to be returned to the script.

For example, when lines of traced LOGON or DELAY statements are encountered while looking for a session screen, and the TRACK statement specifies LINE, the line of trace data is returned to the script in variable field, PTETLINE, and the TRACK statement ends with PTERESP set to the value, LINE.

A returned LINE is a logical record read from a trace dataset that is not a trace title or a part of a traced screen. LINE views non-screen-related data as logical records and returns unedited lines.

Adding Comments and Logic to a Trace Data Set

TRACK STATEMENT and TRACK LINE were designed primarily to make traced LOGON and DELAY statements executable. However, the TRACK statement can handle more non-trace data than just traced LOGON and DELAY statements. Comments and other PSL statements may be added to a format trace dataset using ISPF EDIT, and the modified trace dataset may continue to be used to perform regression testing with analyzer scripts.

However, if the TRACK statement is used to process a format trace dataset, (all CA Enterprise Tester Session Analyzers use the TRACK statement to process format trace datasets), then any data inserted in the dataset must be understood by the TRACK statement.

The TRACK statement interprets the data in a format trace dataset using the following Track Rules:

- Position 1 of a trace dataset record is a RECFM=A printer control character. TRACK recognizes a trace record with a 1 in position 1 as a trace page title of the format generated by the CA Trace facility, and recognizes trace records with +, 0+, or | beginning in record position 1 as lines of a traced screen as generated by Trace.
- Trace records recognized as records generated by CA Enterprise Tester Trace should not be inserted in a trace dataset unless the records are formatted exactly like CA Enterprise Tester trace records and unless any inserted trace screen records are complete screens formatted exactly like those generated by CA Enterprise Tester Trace.
- Records should not be inserted within the set of records comprising a traced screen as generated by CA Enterprise Tester Trace.
- Trace records other than page titles and the records that comprise traced screens are PSL records and must conform to the rules of the CA Enterprise Tester Script Language with the exceptions that follow.
- Position 1 of PSL records must be blank. Positions 2 through 73 of PSL records are interpreted in a manner similar to positions 1 through 72 of CA Enterprise Tester Script Language records.
- PSL statements contained in a trace data set cannot be branched to using GO TO or other PSL statements and therefore PSL records may not have statement labels.
- PSL records may include both types of comments supported by the CA Enterprise Tester Script Language, either comments indicated with an asterisk, *, beginning in PSL record position 2, comments bounded by slash asterisk and asterisk slash, /* comments */ that begin and end anywhere within PSL record positions 2 through 73. PSL statements may be coded in PSL record positions 3 through 73 and may spanas many consecutive records as are required to complete a statement. /* and */ may be inserted that span one or more complete trace screens to comment out the screens that are between /* and the */.
- PSL statement data must begin with a PSL verb that is executable using the PSL EXECUTE statement, and is delimited by the next permissible PSL verb, trace title, or trace screen. The TRACK statement does not perform PSL statement validation and recognizes only permissible PSL verbs. Therefore, PSL statements inserted in a trace dataset should not use actual data values or statement labels that are also PSL verbs.

- However, the TRACK statement does understand PSL literals, and /* */ comments, even when comment delimiters span multiple trace dataset logical records, and is not sensitive to the data contained within the literal quotes or within the /* */ comment delimiters.
- While it is not a rule, it is a good idea to insert PSL statements or comments only before the first screen of a session or before inbound screens. The reason is that if a PSL statement is inserted between consecutive outbound screens, and the dataset is processed using a TRACK statement that specifies both TRANSACTION and STATEMENT, in order to return both the requested transaction screen and the inserted PSL statement in the order that they are found in the trace dataset, TRACK must assume that an outbound EB, (end bracket), screen that precedes the inserted PSL statement is the end of a transaction, and therefore returns the screen before the PSL statement. If an EB screen also follows an inserted statement, a Session Analyzer may get out-of-sync with a live session being analyzed. However, if a Session Analyzer is designed to deal with a PSL statement inserted at such points, the inserted PSL statement may not cause a problem.

Because all PSL statements written to a trace dataset by CA Enterprise Tester conform to the Track Rules listed above, the TRACK statement correctly processes all PSL statements and screens created by CA Enterprise Tester regardless of the options specified on a TRACK statement used to process the trace data.

Whether or not one of the keywords, STATEMENT or LINE is specified on a TRACK statement, the TRACK statement recognizes all trace data that conforms to Track Rules when selecting data requested by a TRACK statement.

For example, when /* */ comments are used to comment out one or more trace screens and a track statement does not specify LINE, the TRACK statement is still aware that the screens are commented out, and skips to the end of the comment before resuming the search for a requested screen or statement.

When a TRACK statement specifies STATEMENT, only specified screens and complete PSL statements are returned to a script, and comments and screens that may be commented out are skipped.

When a TRACK statement specifies LINE, comment lines, including lines of a commented out screen are returned to a script as lines. When a TRACK statement specifies neither LINE or STATEMENT, only specified screens that are not commented out are returned to the script, and all comments, PSL statements or other non-screen trace data is skipped. Refer to PTERESP in PSL Variables for a description of data set access response codes and particular error codes.

Information Returned To a Script in CA Enterprise Tester Variables

During the execution of a TRACK statement, the variable field, PTERESP, is set to indicate the success of the the request. An error handling routine may examine PTERESP to determine the exact cause of an error.

In addition to reading traced screen buffers, the TRACK statement also sets several CA Enterprise Tester defined variables to values that provide information about the screen image that was read.

- PTETPLU is the name of the online system that was specified when the traced session was initiated.
- PTETSLU is the name of the virtual terminal that was used for the traced session.
- PTETMODL the terminal model used for the traced session. PTETMODL may be one of the model keywords, MOD2, MOD3, MOD4or MOD5or may be the name of the VTAM logon mode table entry that was used when the traced session was started.
- PTETDSIZ and PTETASIZ are set to the default and alternate screen sizes that were in use when a traced session was captured. The value of PTETDSIZ or PTETASIZ may be 24X80, 32X80, 43X80or 27X132.

PTETLUDS is set to one of the values, BASICDS or EXTDS and indicates the number of virtual terminal buffers that were used to store the screen image in memory.

- PTETSEQ contains a screen sequence number in the format, #nnnnn, as it appears in the screen title, when processing a trace data set produced by CA Enterprise Tester Release 3.0 or later. When a processing a trace produced by an older release of CA Enterprise Tester, the TRACK statement sets PTETSEQ to the value, #00000.
- PTETDIR is set to the value IN, when an inbound screen image is read from the trace data set. PTETDIR is set to the value OUT, when an outbound screen image is read.
- PTETROWS and PTETCOLS are set to the number of screen rows and columns that are contained in the TRACKED screen.
- PTETSIZE is set to the size of the virtual terminal screen that was returned by a TRACK statement, and is equal to the product of screen rows and columns.
- PTETROW and PTETCOL are set to the row and column number at which the cursor was positioned when the screen was traced.
- PTETAID is set to the name of the AID Key that caused an inbound screen image to be sent to the online system.

- PTETCC is set to the number of messages that contributed to the TRACKED screen. The number is one, 1, for inbound screen images. When a CA Enterprise Tester session is traced, that was driven by a CA Enterprise Tester script that by default or by specification processes only outbound messages that free the terminal keyboard, then only those screen images processed by the script are traced. When more then one message is sent prior to freeing the keyboard, the number of messages is noted in the trace heading. When the screen is read by the TRACK statement, PTETCC is set to the number. (CC in PTETCC comes from the VTAM term chain count.) When a TRACK statement specifies LOGICAL or TRANSACTION and outbound screen images are skipped, PTETCC contains the sum of chain counts for skipped screen images plus the count for the screen image returned.
- PTETSTAT is set to a value that indicates the state of the traced session after the traced screen image was sent or received. PTETSTAT is set to one of the values INBOUND, OUTBOUND or CONTENTION.
- PTETCAPT contains any caption that was found on the bottom border of a traced screen. Screen captions may be specified using the Interactive System TRACE command or Script Language TRACE statement before a screen image is captured.
- PTETRTIM is set when an inbound screen image is read, to the response time of the transaction represented by the previously read inbound screen image. The numeric variable is expressed in hundredths of a second, from the previous inbound screen to the most recently read outbound screen.
- PTETTTIM is set when an inbound screen image is read, to the think time that preceded the inbound screen image just read. The numeric variable is expressed in hundredths of a second, measured from the most recently read outbound screen image, to the inbound screen image just read.
- PTETTOD is set to the trace time-of-day from the top-right border of a traced screen image.
- PTETLINE contains either an executable PSL statement when the TRACK statement keyword, STATEMENT is specified, or a logical record from the trace data set when the TRACK statement keyword, LINE is specified.

TYPE

The TYPE statement is used to operate a CA Enterprise Tester virtual terminal keyboard. This statement has the following format:

```
TYPE
   ( <row column> )
   ( 'literal-data' )
   ( "literal-data" )
   ( &variable-data )
```

```
( TOKEN token )
( CODE code )
( TAB BACKTAB NEWLINE HOME ERASE )
( SELECT | DESELECT )
( TRACKED | OVERLAY )
( ENTER | PF1-24 | PA1-3 |
  PEN | CLEAR | ATTN |  AID &aid-variable )
```

<row column>

Moves the cursor to the specified virtual terminal screen row and column.

literal-data

Literal data, enclosed in single or double quotes, to be typed.

&variable-data

A variable field expression that identifies variable data to be typed.

TOKEN

Identifies the value of the literal or variable that follows, token, as a password token that represents a password to be typed.

CODE

Identifies the value of the literal or variable that follows, code, as a password code that represents a password to be typed.

TAB, BACKTAB, NEWLINE, HOME and ERASE

Keywords that namekeys to be pressed to move the cursor to the next or previous field, to the first field on a subsequent screen row or to the first modifiable screen field, or to erase the remainder of the screen field beginning at the current cursor position.

SELECT and DESELECT

Keywords select or deselect the pen detectable selection or attention field at the current cursor location. The operations simulate operations performed by a 3270 light pen or cursor select key, except that SELECT and DESELECT keywords select or deselect a field as specified regardless of the current value of the field designator character, and do not send data when a pen attention field is selected. Use the PEN keyword to send data that is formatted like the data sent by a light pen or cursor select key when a pen attention field is selected.

TRACKED and OVERLAY

Keywords which describe how the TYPE statement should handle attribute bytes and protected screen data found in data retrieved from previous virtual terminal screen images.

ENTER, PF1-24, PA1-3, PEN, CLEAR , and ATTN

AID keywords (Attention IDentifiers), that send data that is formatted like the data sent when a like-named 3270 key is pressed.

AID

Specifies that the value of the variable that follows, &aid-variable, is an AID keyword that is to be pressed to send data that is formatted like the data sent when the like-named 3270 key is pressed.

The string of data specified by a TYPE statement is processed one data item or keyword at a time. Literal data or data contained in variable fields is keyed into the CA Enterprise Tester screen one character at a time. After each keystroke, the cursor is repositioned as it would be on a real terminal screen, considering screen attributes and screen field wrap processing past the end of a row or the last screen row and column. The modified data tag of the attribute that controls each modified screen field is turned on.

The data represented by a variable field expression specified in a TYPE statement is keyed into the CA Enterprise Tester screen in the same manner as literal data. The variable field expression may specify that an occurrence or a variable field substring be keyed.

Password Security

Tokens identified by the TOKEN keyword, codes identified by the CODE keyword, and TRACKED non-display data are considered to be confidential data and are managed by the active password security system.

Token Security

- Does not allow confidential data to be typed into a display field;
- Converts confidential data and other data typed into non-display fields to token data;
- Hides token data in memory to prevent access using LOG, SET and other PSL statements that have access to virtual terminal display buffers;
- Replaces token data with tokens in recorded or traced session data;
- Replaces token data with nulls in logged session RU data;
- Converts token data to real data, using the user ID that owns the address space, when a TYPE statement sends it to an online system; and
- Converts data sent by a system in modifiable non-display fields and known confidential data in protected non-display fields to token form.

Code Security

- Does not allow confidential data to be typed into a display field.
- Converts confidential data to real data, using the user ID that owns the address space, when it is typed into a non-display field;

- Protects passwords defined in Code Security .
- Replaces passwords with codes in recorded or traced session data. and
- Replaces passwords with nulls in logged session RU data.

Special Function Keyboard Keys

TYPE statement keywords may be used to press special keyboard keys.

- The ERASE keyword simulates the function of the 3270 Erase EOF key. That is, the remainder of the screen field beginning at the current cursor position through the end of the field is cleared to nulls (binary zeros). The field's modified data tag is set to indicate that the field has been modified.
- The HOME keyword moves the cursor to the first unprotected screen field.
- The TAB keyword moves the cursor to the next unprotected screen field that follows the current cursor location.
- The BACKTAB keyword moves the cursor to the beginning of the first unprotected screen field that precedes the current cursor position.
- The NEWLINE keyword moves the cursor to the first unprotected screen field following the screen row on which the cursor is currently positioned.

Cursor Position

The cursor may be positioned at a desired virtual terminal screen location by specifying the row and column at which the cursor is to be placed. The row and column specifications may be absolute numbers or numeric variable fields that contain the screen row and column. The row and column numbers are enclosed in greater than and less than signs that are used like parenthesis.

For example:

"<3 10>" specifies that the cursor is to be positioned at screen row 3 and screen column 10.

"<&row &col>" specifies that the cursor is to be positioned at the screen row and column that is contained in the variable fields, &rowand &col.

3270 Light Pen and Cursor Select Keys

TYPE statement keywords, SELECT and DESELECT perform the same operations on pen detectable selection or attention fields that are performed by 3270 light pen and cursor select keys.

The SELECT keyword operates on the field attribute and designator character of the field in which the virtual terminal cursor is positioned. If the field is pen detectable and the field designator character is null, &, > , or ? , then the field MDT is turned on and a ? designator character is changed to >.

The SELECT keyword only performs operations on valid detectable field attributes and designator characters. A PEN or ENTER AID keyword must be used to generate the appropriate AID after a null or & pen attention field, (or simulated ENTER key), is selected.

The DESELECT keyword operates on the field attribute and designator character of a valid pen selection field in which the virtual terminal cursor is positioned when the DESELECT keyword is processed. If the field is pen detectable and the field designator character is > or ?, then the field MDT is turned off and a > designator character is changed to ? .

The PEN keyword generates a pen aid and sends a 3270 pen datastream to the online system. Like all TYPE statement AID keywords, PEN must be the last argument specified in a TYPE statement or may be specified with a variable that follows the AID keyword.

Attention Identifier Keys

Any of the AID keywords ENTER, CLEAR, PA1 through PA3, PF1 through PF24, or PEN cause a message to be sent to the connected online system that looks like the data stream sent by a real 3270 terminal when a similar key is pressed.

It is not a requirement that an AID key be specified on every TYPE statement. A TYPE statement may be used to simply key data into the virtual terminal screen without sending input to the online system.

If an AID key is not specified, then the TYPE statement completes after all specified data has been keyed. However, a virtual terminal keyboard has states similar to those of a real keyboard.

Virtual Terminal Keyboard State

The state of a virtual terminal keyboard is indicated by values in variable field PTEKB. When data is keyed into a virtual terminal, the state of the keyboard changes to busy.

While PTEKB contains BUSY, subsequent invited online system messages would be rejected, emulating the action of a real terminal. Therefore, a script that keys data should eventually press an AID key to send keyed data.

When an AID key is specified in a TYPE statement, the AID keyword delimits TYPE data. When the AID key is pressed, an appropriate message is sent to the online system and the state of the virtual terminal keyboard is changed to INHIBITED. The TYPE statement normally completes when a message is sent to the terminal that frees the terminal keyboard. When the keyboard is freed, PTEKB is set to FREE, and script execution resumes at the next script statement.

If a script executes a HANDLE OUTBOUND statement to gain access to online system messages that do not free the terminal keyboard, then a TYPE statement completes when any online system response has been received. If the response does not free the keyboard, then the keyboard remains in the inhibited state. While PTEKB contains INHIBITED, a TYPE statement may not be used to key data or to press an AID key, and the script should eventually execute an INVITE statement to receive the next online system message. While

the keyboard is inhibited, a TYPE statement may be used only to send an Attention key signal to the online system. The Attention key is pressed by the TYPE statement keyword, ATTN. The Attention key may be used while the keyboard is BUSY, FREE or INHIBITED. An Attention key signal does not change the keyboard state as other entered messages do. Rather, a TYPE statement that specifies ATTN causes an SNA SIGNAL request to be sent, then behaves as though an INVITE statement has been executed.

If a script executes a HANDLE TIMEOUT statement to limit the amount of time that CA Enterprise Tester will wait for an online system response, then a TYPE statement will complete prematurely if an online system response is not received within the time interval specified on a TIMEOUT statement. When a session times out, the virtual terminal keyboard will be in either an inhibited or locked state. When PTEKB contains LOCKED, a timeout occurred while a session was in a state that disallows any session data traffic. For example, while a session is being passed from one online system to another. While a virtual terminal keyboard is locked, the TYPE statement may not be used. The INVITE statement may be used to resume processing of a timed out session event.

If a script executes a HANDLE TURNAROUND OFF, or HANDLE TURNAROUND AT statement, then the TYPE statement becomes a send-only operation. The TYPE statement will complete after changing the session direction to outbound, and will not wait for a reply. The INVITE statement is subsequently used to receive online system responses.

When a TYPE statement is executed, the state of a session changes when data is sent, when responses are received and when other session events are being processed. The variable field, PTESTATE, contains the current state of a session. CA Enterprise Tester Logical Unit States describes all of the possible states of a session.

TYPE OVERLAY Statement

The TYPE statement OVERLAY keyword specifies that the data to be keyed contains formatted screen data possibly including screen attributes and protected data. The OVERLAY keyword must be the first keyword following the TYPE verb and applies to all data specified in the TYPE statement.

When the TYPE OVERLAY statement is processed, only those characters of the typed data that are not attributes and that correspond to unprotected screen fields are keyed. When a character is keyed, the modified data tag in the attribute that controls the overlaid screen field, is turned on. The cursor is advanced one position after keying or ignoring each typed character.

The TYPE OVERLAY operation may be used to perform cut and paste operations, where entire rows of screen data, including attributes and protected data are remembered in variable fields, then at a later time, the remembered screen field may be keyed over another session screen. The TYPE OVERLAY operation keys only those characters that correspond to modifiable screen fields.

Typing TRACKED Screen Data

The TYPE statement may be used to re-key screen data that was read from a CA Enterprise Tester trace data set using the Script Language TRACK statement. After reading a traced screen, a script may TYPE or otherwise refer to the traced screen by coding the TRACKED adjective prior to a screen field expression. TYPE statement support for TRACKED data is designed to support regression testing scripts that automatically repeat and compare previously traced CA Enterprise Tester sessions. Refer to Regression Testing for complete documentation of CA Enterprise Tester regression testing capabilities.

When TRACKED data is typed over a CA Enterprise Tester session screen, only data from the traced screen that is contained in an unprotected and modified trace screen field is re-keyed. When a character is re-keyed, the modified data tag in the attribute that controls the re-keyed screen field, is turned on. After re-keying each character, the cursor is advanced one position. When a TRACKED screen field has been re-keyed without error, the variable fields, PTEKE and PTEKEL, are set to 0.

When TRACKED data is re-keyed, a keying error is recognized when a character that is to be keyed can not be keyed because the cursor is positioned in a protected field of the active session screen. When a keying error is encountered, re-keying of the TRACKED screen field is terminated, the variable field, PTEKE, is set to the number of the first position within the specified trace screen field that could not be re-keyed, and the variable field, PTEKEL, is set to the net length of the specified trace screen field beginning at the first position that could not be re-keyed. A keying error terminates only the re-keying of the TRACKED screen field. If additional TYPE statement specifications follow the TRACKED screen field, then they are processed normally. A script may detect keying errors by typing only one TRACKED screen field at a time, and by testing the value in PTEKE following execution of the TYPE statement.

Also in support of session repeat and compare scripts, the TYPE statement allows an AID key to be specified using a variable field. When a traced screen is read using the TRACK statement, the name of the AID key that caused a traced inbound screen, is placed in the variable field, PTETAID. A script may re-key the traced AID key using the TYPE statement AID keyword. The following statement positions the active session cursor as it was position when the traced screen was sent to an online system, and presses the traced AID key:

```
TYPE <&PTETROW,&PTETCOL> AID &PTETAID
```

UPDATE

The UPDATE statement is used to replace the record in a VSAM Key Sequenced Data Set (KSDS), whose key exactly matches the key of a record created by the UPDATE statement. The UPDATE statement creates the new record by concatenating a string of one or more data items together to form a new record, then replaces the record in the data set whose key matches the key of the constructed record. This statement has the following format:

```
UPDATE ddname FROM string
     ( NOTFOUND routine )
     ( ERROR routine | * )
```

ddname

The name of a data definition, (DD), statement that identifies the data set in which a record is to be updated; ddname may be the actual DD name or may be expressed as a variable field that contains the DD name; ddname must have been previously opened by executing an OPEN statement; the DD name specified may not begin with the characters, PTE.

FROM

A required keyword and indicates that a string of data items follows.

String

A list of literal fields, variable fields and screen fields that are to be concatenated together to construct the record that is to replace a record with a matching key in the VSAM KSDS.

Examples

```
UPDATE VSAMKSDS FROM &RECORD
UPDATE VSAMKSDS FROM &KEY &DATA
                NOTFOUND ADDIT
```

CA Enterprise Tester provides automatic unusual condition processing when an optional condition keyword is specified. Each condition keyword is followed by the name of a script routine that is to receive control if the condition is encountered.

A condition handling routine may be expressed as a statement label or number, as a variable field that contains a statement label or number, or may be the special symbol, asterisk, *, to specify that execution is to be resumed at the next script statement. Condition keywords supported by the UPDATE statement are NOTFOUND and ERROR.

- NOTFOUND routine identifies a script routine that is to receive control if a record is not found in the VSAM KSDS that has a key that matches the key of the record specified by the UPDATE statement.
- ERROR routine identifies a script routine that is to receive control when an error or other unhandled condition is encountered during the execution of an UPDATE statement. IF ERROR is not specified on an UPDATE statement, an ERROR routine specified on the associated OPEN statement will be used.

The UPDATE statement constructs a record by concatenating the data items listed, following the FROM keyword. For fixed length data sets, the constructed record is padded with blanks or is truncated, if necessary, to form a record of the required length. For variable length data sets, constructed records contain only the specified data. If necessary, variable length records are truncated to the maximum logical record length for the data set. The UPDATE statement then replaces the VSAM record whose key matches the key of the record constructed from UPDATE statement data.

The UPDATE statement can be used only to replace an existing record. A NOTFOUND condition occurs if the data set does not contain a record with a matching key. If NOTFOUND handling is not specified, a not-found condition is considered to be an error and may be handled at an ERROR routine specified on the UPDATE or OPEN statements. The WRITE statement may be used to insert a new record into a VSAM KSDS.

During the execution of the UPDATE statement, the variable field, PTERESP, is set to indicate the success of the request. An error handling routine may examine PTERESP to determine the exact cause of an error. Refer to PTERESP in CA Enterprise Tester Defined Variables for a description of data set access response codes and particular error codes.

Refer to Data Set Access for an overview level description of CA Enterprise Tester data set access capabilities.

WAIT

The WAIT statement delays script execution for a specified period of time. This statement has the following format:

```
WAIT hhmmssth
```

Hhmmssth

The wait delay interval expressed in hours (hh), minutes (mm), seconds (ss) and tenths and hundredths of seconds (th). A WAIT interval may be expressed using a character variable field containing data in the format, hhmmssth.

Example:

```
WAIT  1500    /*fifteen seconds*/
```

The statement may be used to control monitoring frequency or to slow down a script to simulate an operators transaction entry speed.

When a script is executing under the control of the CA Enterprise Tester Interactive System and a user enters a WAIT or STEP command to wait for session activity, then the WAIT interval may be suppressed so that the terminal operator does not have to wait for the interval to expire. CA Enterprise Tester provides an option on the Interactive System Options Panel which controls the way the WAIT statement is processed. The CA Enterprise Tester Options Panel is described in Options and Controls.

WRITE

The WRITE statement is used to serially add records to a sequential data set, to sequentially load an empty VSAM Key Sequenced Data Set (KSDS) and to randomly insert new records into a VSAM KSDS. The WRITE statement creates a record by concatenating a string of one or more data items together and writes the constructed record to a specified user data set. This statement has the following format:

```
WRITE ddname FROM string
   ( DUPKEY routine )
   ( ERROR routine | * )
```

ddname

The name of a data definition, (DD), statement that identifies the data set to which a record is to be written; ddname may be the actual DD name or may be expressed as a variable field that contains the DD name; ddname must have been previously opened by executing an OPEN statement; the DD name specified may not begin with the characters, PTE.

FROM

A required keyword and indicates that a string of data items follows.

String

A list of literal fields, variable fields and screen fields that are to be concatenated together to construct the record to be written.

Examples

```
WRITE SYSUT2 FROM &RECORD
WRITE MSGLOG FROM
    "RESPONSE IS '" <24,1,80> "'"
```

CA Enterprise Tester provides automatic unusual condition processing when an optional condition keyword is specified. Each condition keyword is followed by the name of a script routine that is to receive control if the condition is encountered.

A condition handling routine may be expressed as a statement label or number, as a variable field that contains a statement label or number, or may be the special symbol, asterisk, * , to specify that execution is to be resumed at the next script statement. Condition keywords supported by the WRITE statement are DUPKEY and ERROR.

- DUPKEY routine identifies a script routine that is to receive control if the key of a record written to a VSAM KSDS, matches the key of an existing record in the data set.
- ERROR routine identifies a script routine that is to receive control when an error or other unhandled condition is encountered during the execution of a WRITE statement. If ERROR is not specified on an WRITE statement, an ERROR routine specified on the associated OPEN statement will be used.

The WRITE statement constructs a record by concatenating the data items listed following the FROM keyword. For fixed length data sets, the constructed record is padded with blanks or is truncated, if necessary, to form a record of the required length. For variable length data sets, constructed records contain only the specified data. If necessary, variable length records are truncated to the maximum logical record length for the data set.

Records written to a sequential data set and records written to an empty VSAM KSDS in ascending key sequence are added sequentially to the end of the data set. Records written to a non-empty VSAM KSDS or records written out-of-sequence are inserted randomly.

The WRITE statement can be used only to add new records to a data set. The DUPKEY condition is raised if an attempt is made to WRITE a record to a VSAM KSDS, whose key exactly matches the key of a record already in the data set. If DUPKEY handling is not specified, a duplicate key condition is considered to be an error and may be handled at an ERROR routine specified on the WRITE or OPEN statements. The UPDATE statement may be used to replace a record that has a matching key.

During the execution of the WRITE statement, the variable field, PTERESP, is set to indicate the success of the request. An error handling routine may examine PTERESP to determine the exact cause of an error.

Refer to PTERESP in CA Enterprise Tester Defined Variables for a description of data set access response codes and particular error codes, and to Data Set Access for an overview level description of CA Enterprise Tester data set access capabilities.

WTO

When a WTO statement is executed in a script running as a batch job step, then an assembler WTO macro is issued to process the message using default route and descriptor codes specified when CA Enterprise Tester was installed, or using route and descriptor codes specified using WTO statement keyword parameters.

When a script is played using the CA Enterprise Tester Interactive System, then the WTO statement writes a message to the Message Log that is presented to the terminal user via the Monitor Panel Message Log. This statement has the following format:

```
WTO message
   ( ROUTCDE codes )
   ( DESC descriptors )
   ( NODOM )
   ( NOLIMIT )
   ( NOWRAP )
```

message

A list of literals, variable fields and screen fields to be concatenated to form the message to be written. By default, if the total length of a message exceeds 50 characters, the message is broken into 50 character segments.

ROUTCDE codes

An optional parameter that may be used to override installation specified default message routing codes.

DESC descriptors

An optional parameter that may be used to override installation specified default message descriptor codes.

NODOM

An optional parameter that may be used to specify that the WTO message is not to be deleted following a subsequent WTOR nor at the end of a batch job.

NOLIMIT

An optional parameter that may be used to override installation specified system limits on the number of consecutive WTO statements which may be issued without a WTOR statement.

NOWRAP

An optional parameter that may be used to override default logic which causes messages longer than 50 characters to be sent as multiple messages. NOWRAP will cause a single message to be sent. Message data which exceeds 125 characters is discarded. (The system WTO message length limit is 125 characters.)

Example:

```
WTO 'UNABLE TO LOGON TO ' &PTEPLU
    ' FROM ' &PTESLU
```

When executed under the control of the CA Enterprise Tester Interactive System, WTO messages are logged for examination by the terminal user. The terminal user may view WTO messages issued by a script via the Message Log presented on the Interactive System Monitor Panel.

When a script is executed as a batch job, the WTO statement may be used to log messages in the z/OS job log or to send important messages to a console operator. A copy of each WTO message is always included in the CA Enterprise Tester Log, specified via the PTELOG DD statement. The destination and significance of a WTO messages are specified using route and descriptor codes.

Using the WTO statement ROUTCDE parameter, a WTO statement may specify one or more codes that specify the destination of the message.

For example:

```
WTO 'SEND THIS TO PROGRAMMER' ROUTCDE 2 11
```

If route codes are not specified, installation specified default route codes will be used. The meaning of route codes is defined by IBM in documentation of the assembler WTO macro statement. Refer to IBM publications for full details.

The following table lists a brief description of the intended purpose of each route code.

Code	Description	Code	Description
1	Master console action	9	System security
2	Master console information	10	System error/maintenance
3	Tape pool	11	Programmer information
4	Direct access pool	12	Emulators
5	Tape library	13	Reserved for customer
6	Disk library	14	Reserved for customer
7	Unit record pool	15	Reserved for customer
8	Teleprocessing control	16	Reserved for future expansion

Using the WTO statement DESC parameter, you may describe the significance of a message. For example:

```
WTO 'Check CICS1 system status' ROUTCDE 2 DESC 2
```

If descriptor codes are not specified, installation specified default descriptor codes will be used. The meaning of descriptor codes is defined by IBM in documentation of the assembler WTO macro statement. Refer to IBM publications for full details.

The following table lists a brief description of the significance each descriptor code.

Code	Description	Code	Description
1	System failure	7	Application program/processor
2	Immediate action required	8	Out-of-line message
3	Eventual action required	9	Operator request
4	System status	10	Dynamic status displays
5	Immediate command response	11	Critical eventual action requested
6	Job status	12-16	Reserved for customer

Inappropriate or erroneous use of the WTO statement and routing and descriptor codes may cause significant system problems. For example, if a script gets into a loop that contains a WTO with ROUTCDE 2 and DESC 2, and floods the system console with messages, console lockout is possible. To allow some system level control of this possibility, an installation option allows a limit to be placed on the number of consecutive WTO statements that may be issued without an intervening WTOR. By default, the limit is ten. Any previous WTO messages are deleted when a response is issued to a WTOR message, unless the NODOM option was specified. If the NODOM option was specified, then previous messages are not deleted.

The WTO statement NOLIMIT option may be used to override any specified system limit, allowing any number of consecutive WTO statements to be issued. A WTO statement that specifies NOLIMIT is not deleted when a response is received to a WTOR

If in doubt, you should consult system programming personnel at your installation about appropriate use of route and descriptor codes and defaults in effect at your installation.

WTOR

The WTOR statement allows a script to send a message to and receive a response from a console operator or terminal user.

When a WTOR statement is executed in a script running as a batch job step, then an assembler WTOR macro is issued to process the message using default route codes specified when CA Enterprise Tester was installed, or using route codes specified via the WTOR ROUTCDE parameter. The WTOR macro writes a message to the default or specified destination, then waits for an operator reply.

When a WTOR statement is executed by a script executing under the control of the CA Enterprise Tester Interactive System, the message is written to the Message Log that is displayed on the Interactive System Monitor Panel, and the script is stopped until the terminal user replies to the message using an Interactive System REPLY command. This statement has the following format:

```
WTOR message
  ( ROUTCDE codes )
  ( DESC descriptors )
  ( NOWRAP )
```

message

A list of literals, variable fields and screen fields to be concatenated to form the message to be written. By default, if the total length of a message exceeds 50 characters, the message is broken into 50 character segments.

ROUT Decodes

An optional parameter that may be used to override installation specified default message routing codes.

DESC descriptors

An optional parameter that may be used to override installation specified default message descriptor codes.

NOWRAP

An optional parameter that may be used to override default logic which causes messages longer than 50 characters to be sent as multiple messages. NOWRAP will cause a single message to be sent. Message data which exceeds 121 characters is discarded. (The system WTOR message length limit is 121 characters.)

Example

```
WTOR 'REPLY "RETRY" OR "ABEND"'
```

When a reply is received from the console operator or from the Interactive System terminal operator, the variable field, PTEREPLY, is set to the value entered. PTEREPLY may be examined by the script to determine what action to take.

When a reply is received to a message written to the system console, then any previous messages sent to the console using WTO statements are automatically deleted, unless the NODOM option was specified when the WTO was issued. If the NODOM option was specified, then previous messages are not deleted.

A copy of the WTOR message and operator response is always written to the CA Enterprise Tester Log, specified via the PTELOG DD statement for batch jobs and the Monitor Panel Message Log for scripts executed using the CA Enterprise Tester Interactive System.

When a WTOR is issued by a script running under the control of the CA Enterprise Tester Interactive System, then any route code specifications are ignored. For batch jobs, the destination of WTOR messages may be specified using route codes as follows:

Using the WTOR statement ROUTCDE parameter, a WTOR statement may specify one or more codes that specify the destination of a message.

For example:

```
WTOR 'Reply "retry" or "abend"' ROUTCDE 2
```

If route codes are not specified, installation specified default route codes will be used. The meaning of route codes is defined by IBM in documentation of the assembler WTOR macro statement. Refer to IBM publications for full details.

Descriptor codes are meaningless in an z/OS WTOR macro statement. Descriptors specified on a Script Language WTOR statement apply only to the WTO macros that are issued when a WTOR message exceeds 50 characters.

WTO and WTOR statement route and descriptor codes have the same meaning and values.

Chapter 7: PSL Variables

This chapter documents special variables defined by CA Enterprise Tester for reference by a script.

- PSL Variables

PSL Variables

A variable name that begins with the characters PTE is reserved. CA Enterprise Tester defines The content and use of these key elements. CA Enterprise Tester defined variables can not be modified by a script using a SET statement, but can otherwise be referenced wherever user-defined variables can be referenced.

CA Enterprise Tester uses defined variables to provide information to a script. When referred to in a Script Language statement, CA Enterprise Tester defined variables and user defined variables are prefixed by an ampersand, &, to distinguish variable references from other statement elements.

The following list defines the purpose and use of CA Enterprise Tester defined variables.

PTEARGC

When the CONTAINS condition is used in a conditional statement to search a virtual terminal screen field for an argument, then PTEARGC is set to the screen column at which the argument was found. If the argument is not found, PTEARGC is set to zero. PTEARGC is a numeric variable.

PTEARGI

When the CONTAINS condition is used in a conditional statement to search a variable field or a variable field substring, then PTEARGI is set to the position in the variable field at which the argument was found. If the argument is not found, PTEARGI is set to zero. PTEARGI is a numeric variable.

PTEARGR

When the CONTAINS condition is used in a conditional statement to search a virtual terminal screen field for an argument, then PTEARGR is set to the screen row at which the argument was found. If the argument is not found, PTEARGR is set to zero. PTEARGR is a numeric variable.

PTEASIZ

When a session is started, PTEASIZ is set to one of the values 24X80, 32X80, 43X80 or 27X132, and indicates the alternate screen size established for the session. PTEASIZ is a character variable.

PTECC

After the execution of a TYPE or INVITE statement, PTECC is set to the number of outbound chains sent to CA Enterprise Tester by the connected online system. CC means chain count. A chain is a series of messages which collectively construct a logical message. If PTECC is zero, following the execution of an INVITE statement, then a message from the online system was not waiting to be received. PTECC is a numeric variable.

PTECOL

During the execution of a TYPE statement, the cursor is repositioned as data is keyed into a CA Enterprise Tester screen image. When an online system sends a terminal message, the message can reposition the cursor. Whenever the cursor is repositioned, PTECOL and PTEROW are set to the CA Enterprise Tester screen image column and row at which the cursor is positioned. PTECOL and PTEROW are numeric variables.

PTECOLS

If a session exists between a script and an online system, this field contains the current number of columns in the virtual terminal screen, otherwise PTECOLS is zero. Screen size is always determined by the online system through the use of the 3270 erase-write and erase-write-alternate commands. PTECOLS is a numeric variable.

PTEDD

At the start of script execution and during the execution of a Script Language DATE statement, the fields, PTEMM, PTEDD, and PTEYY, are set to the current month, day and year. Each field is a two-position character variable.

PTEDEL

Following the execution of a Script Language DELETE statement, PTEDEL is set to the number of records that were deleted during execution of the statement. The number can be greater than one when a DELETE statement specifies a generic VSAM key.

PTEDSIZ

When a session is started, PTEDSIZ is set to one of the values 24X80, 32X80, 43X80 or 27X132, and indicates the default screen size established for the session. PTEDSIZ is a character variable.

PTEHR

At the start of script execution and during the execution of a Script Language TIME statement, the variables, PTEHR, PTEMIN, and PTESEC, are set to the current hour, minute and second. Each field is a two position character variable. Refer also to the variable, PTETIME.

PTEJOB

When script execution begins, PTEJOB, is set to the name of the job or TSO address space that is being used to execute the CA Enterprise Tester script.

PTEKB

PTEKB contains one of the following words, and indicates the current state of the virtual terminal keyboard.

Note: Keyboard state names that have lower alphabetic values are more free and names with higher values are locked tighter. Any TYPE statement is valid when PTEKB is less than or equal to FREE. No TYPE statement is valid when PTEKB is greater than or equal to LOCKED.

- OFFLINE means a session is not active.
- LOCKED means that a session has timed out in a state that does not permit the use of the TYPE statement.
- INHIBITED means that a session has timed out or is outbound and the TYPE statement can be used only to TYPE ATTN.
- FREE means that the keyboard is free and all TYPE statements features can be used.
- BUSY means that the keyboard is free and all TYPE statement features can be used, and that the TYPE statement has been used to enter data, but that an AID key has not yet been pressed.

PTEKE

When a portion of a traced screen image that was read from a CA Enterprise Tester trace data set, that is, TRACKED data, is keyed into an active CA Enterprise Tester Session screen image, PTEKE is set to zero if the re-keying operation completed without error. If the format of the TRACKED data conflicts with the format of the session screen, then PTEKE is set to the number of the first position within the TRACKED data field that could not be re-keyed.

PTEKEL

When a portion of a traced screen image that was read from a CA Enterprise Tester trace data set, that is, TRACKED data, is keyed into an active CA Enterprise Tester Session screen, PTEKEL is set to zero if the re-keying operation completed without error. If the format of the TRACKED data conflicts with the format of the session screen, then re-keying operation is stopped at the first TRACKED field position that could not be re-keyed and PTEKEL is set to the number of TRACKED characters that were not re-keyed.

PTELINE

If a script executes a HANDLE condition AT label statement, and the condition arises, then before going to the specified label, PTELINE is set to the line number of the script statement at which the condition was detected.

PTENETL

When the CONTAINS keyword is used in a conditional statement to search for an argument in any field of data and the argument is found, and other data follows the argument in the specified field of data, then PTENETL is set to the length of the net field data that was not searched. If the argument is not found or no data follows the argument in the searched field, then PTENETL is set to zero. PTENETL is a numeric variable.

PTENETR

When the CONTAINS keyword is used in a conditional statement to search a virtual terminal screen field for an argument, and the argument is found, and other data follows the argument in the searched screen field, then PTENETR is set to the row of the net screen field data. If the argument is not found or no data follows the argument in the searched screen field, then PTENETR is set to zero. PTENETR is a numeric variable.

PTENEXT

If a script executes a HANDLE condition AT label statement, and the condition arises, then before going to the specified label, PTENEXT is set to the line number of the script statement that would have been executed next if the condition had not occurred.

PTEPGM

PTEPGM allows a script to determine if it is being attended by a user or if it is running in an unattended mode. When a script is played using the CA Enterprise Tester Interactive System Play or Analyze Panel, PTEPGM is set to the value 'PTEIS'. When a script is played using batch JCL or by calling program PTEBS, PTEPGM is set to the value 'PTEBS'.

PTEPLU

After the execution of a LOGON statement, PTEPLU contains the name of the online system that was logged onto. The value of PTEPLU is the name of the online application supplied by the script via a LOGON statement. PTEPLU can be modified by an installation-written Communications Exit Program during LOGON statement processing. PTEPLU is also updated in the event that a session is passed by an online system to another online system.

PTEPORT

If a script is executed using the CA Enterprise Tester Interactive System, PTEPORT is set to the number of the port on which the script is executing. Currently, if a script is executed as a batch job, PTEPORT is set to zero. PTEPORT is a numeric variable field.

Important! Do not use PTEPORT to make decisions about whether a script is running in batch or under TSO based on this variable. A planned multi-session feature will allow scripts to be attached to CA Enterprise Tester ports, even when they are running as a batch job!

Usually, a TSO address space job name is the same as the user ID used to sign on to TSO, so if this convention is followed in your system, you can examine CA Enterprise Tester Variables, PTEJOB and PTEUSER, and if they are equal, then the script is executing under TSO.

PTERC

Following the execution of a Script Language CALL statement, PTERC is set to the return code returned by the called program.

PTEREPLY

The reply that is entered by a system console operator in response to a Script Language WTOR statement is placed in the PTEREPLY variable for examination by the script. If a HANDLE LOGONFAIL ON statement has been executed and a LOGON statement subsequently fails, and the operator replies DOWN or REJECT to the resulting prompt, then PTEREPLY is set to the operator's reply.

PTERESP

Following the execution of any of the Script Language data set access statements, ALLOCATE, OPEN, READ, TRACK, WRITE, UPDATE, DELETE, CLOSE or FREE, PTERESP is set to a four-character code that indicates the success of the request or the cause of an abnormal condition or error.

The code contained in PTERESP can be one of the following values:

- 0000, indicating that the statement was successful;
- Either a VSAM return and error code, expressed in hexadecimal characters;
- A system abend code that begins with the character S, which is followed by the hexadecimal three-character abend code;

- A four-character code describing the type of data returned as the result of a PSL TRACK statement.

When the TRACK statement keyword, STATEMENT or LINE is used to return PSL statements or logical records read from a trace dataset, PTERESP is set to the value, LINE, and the data is returned in the variable, PTETLINE.

An end-of-file condition encountered when reading a sequential data set is expressed as a VSAM return code.

Complete documentation on data set access abend and error codes returned in PTERESP can be found in IBM-supplied documentation of System Completion Codes or in documentation of VSAM return and error codes.

The following is a list of common VSAM return and error codes following an OPEN request:

0000

The OPEN completed successfully.

0474

The data set was not properly closed or is open on another CPU. OPEN completes if WARNING is handled.

08A0

Probable open of an empty VSAM data set for INPUT.

The following is a list of common VSAM return and error codes while accessing a data set:

0000

The request completed successfully.

0804

End of file encountered during READ.

0808

An attempt was made to WRITE a duplicate key.

0810

Record not found for READ, UPDATE, or DELETE.

081C

VSAM data set is full and can not be extended.

The following is a list of common DCB abends while processing sequential data sets:

S013

A PDS member opened for input does not exist.

SD37

A WRITE failed because the data set is full.

PTEROW

During the execution of a Script Language TYPE statement, the cursor is repositioned as data is keyed into a CA Enterprise Tester Screen. When an online system sends a terminal message, the message can reposition the cursor. Whenever the cursor is repositioned, PTEROW and PTECOL are set to the CA Enterprise Tester virtual terminal screen image row and column at which the cursor is positioned. PTEROW and PTECOL are numeric variables.

PTEROWS

If a session exists between the script and an online system, this field contains the current number of rows in the virtual terminal screen, otherwise PTEROWS is zero. Screen size is always determined by the online system through the use of the 3270 erase-write and erase-write-alternate commands. PTEROWS is a numeric variable.

PTERSENS

If VTAM is unable to start a session requested by a LOGON statement, or if an active session fails or data sent to an online system is rejected during the execution of TYPE statement or other PSL session statement, PTERSENS is set to the eight character sense code received from VTAM or the online system. Sense codes are defined by SNA and indicate the cause of a session failure or other negative response. If a LOGON fails, or a session fails unexpectedly, a script can analyze the contents of PTERSENS to determine the cause. For example, 08570002 means the PLU is not active and 087D0001 indicates a session services path error, possibly resulting from an attempt to start a session with an undefined PLU. General information on sense codes, can be found in SNA Sense Codes.

PTERTIME

Following the execution of each TYPE statement and each INVITE statement that receives a message from an online system, PTERTIME is set to the response time of the most recently entered transaction. Each response time is measured from the time of the last terminal input, that was sent using a TYPE statement, to the time when the most recent message was received from the online system. PTERTIME is a numeric variable and contains the response time interval expressed in hundredths of a second.

PTESEC

At the start of script execution and during the execution of a Script Language TIME statement, the variables, PTEHR, PTEMIN and PTESEC are set to the current hour, minute and second. Each field is a two position character variable. Refer also to the variable, PTETIME.

PTESENSE

While executing a Script Language TYPE or INVITE statement, if CA Enterprise Tester sends a negative response to a request received from an online system, then PTESENSE is set to the four character SNA sense code that was sent with the negative response. The sense code indicates the reason that the online system request was rejected. The value, '0000', indicates that a negative response was not sent. General information on sense codes, can be found in SNA Sense Codes.

PTESIZE

While a session is active, PTESIZE is the size of the virtual terminal screen (rows x columns). PTESIZE is a numeric variable.

PTESLU

After the execution of a Script Language LOGON statement, PTESLU contains the name of the virtual terminal from which the script has logged on. This name can be supplied by the script via the LOGON statement. If a virtual terminal name is not specified, then one will be assigned automatically.

PTESTATE

This field contains a name that describes the current state of a CA Enterprise Tester session.

- OFFLINE indicates that a session is not active. If PTESTATE contains any value other than OFFLINE, then a session is active. A session is normally in one of the following session states:
- ONLINE indicates that during the execution of the previous TYPE or INVITE statement, that the online system passed the session to a different online system or that the online system sent an SNA CLEAR command to stop session traffic followed by an SDT to restart data traffic. The state will exist only until the next TYPE or INVITE is executed. If PTESTATE is ONLINE, then the session is also in the CONTENTION state.
- CONTENTION indicates that the session is currently between brackets, and that either the CA Enterprise Tester Script or the online system can send the next message (initiate the next bracket) on the session. INVITE and/or HANDLECONTENTION statements can be used to control bracket initiation while in the contention state.
- INBOUND indicates that the session is currently in-bracket and inbound, meaning that the next message sent on the session must be sent by the CA Enterprise Tester Script to the online system, and that the online system will wait for the message.

- OUTBOUND indicates that the session is currently in-bracket and outbound, meaning that the next message sent on the session must be sent by the online system to the virtual terminal. A script can not execute a TYPE statement while in the outbound state, but rather, must receive the next message from the online system by issuing an INVITE statement. A script can receive messages that continue the outbound state only when a HANDLE OUTBOUND statement has been issued.

If HANDLE TIMEOUT and TIMEOUT statements are used to limit the amount of time that CA Enterprise Tester will wait for the completion of an in-progress session event, and a session times out after waiting the specified limit, then several other session states are possible. A complete list of all possible session states is contained in CA Enterprise Tester Logical Unit States.

PTESTEP

When execution of a script begins, PTESTEP is set to the name of the job step and JCL procedure step under which CA Enterprise Tester is being invoked. PTESTEP is a sixteen-character field.

PTETAID

When an inbound screen image is read from a previously created CA Enterprise Tester trace data set using the TRACK statement, PTETAID is set to the name of the AID Key that caused the traced screen to be sent to the online system. PTETAID will be set to one of the values, ENTER, CLEAR, PF1 through PF24, PA1 through PA3, or ATTN.

PTETASIZ

When a screen image is read from a CA Enterprise Tester trace data set using the TRACK statement, PTETASIZ is set to one of the values 24X80, 32X80, 43X80or 27X132, and indicates the alternate screen size available during the traced session. PTETASIZ is a character variable.

PTETCAPT

The screen caption contained on a traced screen image read by the TRACK statement is extracted and stored in PTETCAPT. A null value is returned when a screen image has no caption.

PTETCC

When a traced screen image is read by the TRACK statement, PTETCC is set to the number of messages that contributed to the traced screen. For inbound screen images, the number is one. For outbound screens, the number can be greater than one when a CA Enterprise Tester script that by default or by specifying HANDLE OUTBOUND ON, receives messages only when a message frees the terminal keyboard creates a trace. While HANDLE OUTBOUND ON is active CA Enterprise Tester traces only those screen images presented to the CA Enterprise Tester Script and notes the total number of outbound message in the trace heading.

PTETCOL

When the Script Language TRACK statement reads a traced screen image, PTETCOL and PTETROW are set to the screen column and row at which the cursor was positioned when the screen image was written to the trace data set.

PTETCOLS

When the Script Language TRACK statement reads a traced screen image, PTETCOLS and PTETROWS are set to the number of screen columns and rows contained in the traced screen image.

PTETDIR

When the Script Language TRACK statement reads a traced screen image, PTETDIR is set to the value IN, when an inbound screen image is read from the trace data set. PTETDIR is set to the value OUT, when an outbound screen image is read.

PTETDSIZ

When a screen image is read from a CA Enterprise Tester trace data set using the Script Language TRACK statement, PTETDSIZ is set to one of the values 24X80, 32X80, 43X80or 27X132, and indicates the default screen size available during the traced session. PTETDSIZ is a character variable.

PTETIME

At the start of script execution and during the execution of a Script Language TIME statement, the variables PTEHR, PTEMIN and PTESEC are set to the current hour, minute and second. The variable PTETIME will be set to a 20 character string of the form, yyyymmddhhmmssthmiju, where the format characters represent:

- yyyy- year
- mm- month
- dd- day
- hh- hour
- mm- minutes
- ss- seconds
- th- tenths and hundredths of seconds
- m- millisecond
- i- ten-thousandths of seconds
- j- hundred-thousandths of seconds
- u- micro-second.

TIME variables are automatically set at the beginning of script execution and are updated only when the TIME statement is executed.

PTETLINE

When a screen image is read from a CA Enterprise Tester trace data set using the TRACK statement keyword, STATEMENT, if a PSL statement such as LOGON or DELAY is the next data found in the dataset being tracked, the entire PSL statement is returned in variable field, PTETLINE. A returned statement is left-justified, with the PSL verb beginning in position 1 of PTETLINE, and all insignificant blanks between statement keywords and data items, and any /* comments */ coded within the PSL statement are removed.

A statement returned by TRACK can be executed using the PSL EXECUTE statement.

PTETLUDS

When a screen image is read from a CA Enterprise Tester trace data set using the Script Language TRACK statement, PTETLUDS is set to one of the values BASICDS or EXTDS, and indicates the 3270 data stream level of the traced screen image.

PTETMODL

When the Script Language TRACK statement reads a traced screen image, PTETMODL is set to the model of the terminal that was used for the traced session. PTETMODL can contain one of the symbolic mode table names MOD2, MOD3, MOD4or MOD5, or can contain that actual name of the VTAM logon mode table entry that was used when the traced session was initiated.

PTETPLU

When a traced screen image is read by the Script Language TRACK statement, PTETPLU is set to the name of the online system that was specified when the traced session was initiated.

PTETROW

When the Script Language TRACK statement reads a traced screen image, PTETROW and PTETCOL are set to the screen row and column at which the cursor was positioned when the screen image was written to the trace data set.

PTETROWS

When the Script Language TRACK statement reads a traced screen image, PTETROWS and PTETCOLS are set to the number of screen rows and columns contained in the traced screen image.

PTETRTIM

When the second or subsequent inbound screen image is read from a CA Enterprise Tester trace data set using the Script Language TRACK statement, PTETRTIM is set to the response time of the transaction represented by the previously read inbound screen image. The response time is measured from the time stamp contained in the previous inbound screen image, to the time stamp contained in the most recently read outbound screen image. PTETRTIM is a numeric variable and contains the response time interval expressed in hundredths of a second.

PTETSEQ

When the Script Language TRACK statement reads a traced screen image, PTETSEQ contains a screen sequence number in the format, #nnnnn, as it appears in the screen title, when processing a trace data set produced by CA Enterprise Tester Release 3.0 or later. When a processing a trace produced by an older release of CA Enterprise Tester, the TRACK statement sets PTETSEQ to the value, #00000 .

PTETSIZE

When a screen image is read from a CA Enterprise Tester trace data set using the Script Language TRACK statement, PTETSIZE is set to the size of the screen image (rows x columns). PTETSIZE is a numeric variable.

PTETSLU

When a traced screen image is read by the Script Language TRACK statement, PTETSLU is set to the name of the virtual terminal that was used for the traced session.

PTETSTAT

When the Script Language TRACK statement reads a traced screen image, PTETSTAT is set to a value that indicates the state of the traced screen after the traced screen was sent or received. Possible values are INBOUND, OUTBOUND or CONTENTION. The meaning of these state names is documented in CA Enterprise Tester.

PTETTIME

Following the execution of each Script Language TYPE statement, PTETTIME is set to the think time that preceded the message sent to the online system by the TYPE statement. Each think time is measured from the receipt of the most recent online system message that preceded the TYPE statement, to the time that the message specified by the TYPE statement was sent to the online system. PTETTIME is a numeric variable and contains the think time interval expressed in hundredths of a second.

PTETTOD

When a screen image is read from a CA Enterprise Tester trace data set using the Script Language TRACK statement, PTETTOD contains the trace time-of-day that is obtained from the top-right border of the traced screen image.

PTETTTIM

When the second or subsequent inbound screen image is read from a CA Enterprise Tester trace data set using the Script Language TRACK statement, PTETTTIM is set to the think time that preceded the inbound screen image. The think time is measured from the time stamp contained in the previously read outbound screen image, to the time stamp contained in the inbound screen image just read by the TRACK statement. PTETTTIM is a numeric variable and contains the think time interval expressed in hundredths of a second.

PTEUSER

PTEUSER is set to the user ID of the person who submitted or owns the job or TSO address space in which a CA Enterprise Tester script is executing. PTEUSER is a character variable field.

PTEYY

At the start of script execution and during the execution of a Script Language DATE statement, the fields, PTEMM, PTEDD and PTEYY, are set to the current month, day and year. Each field is a two-position character variable.

Chapter 8: Testing

This chapter describes various ways in which CA Enterprise Tester can automate tasks related to testing of new and modified application systems, including advanced techniques for regression testing, and multiple session online system tests.

- Introduction
- Regression Testing
- Multiple Session Testing

Introduction

Because CA Enterprise Tester provides the capability to automate tasks performed on 3270-family terminal devices, and can provide a theoretically unlimited number of scripts operating on a theoretically unlimited number of virtual terminals, testing in a data processing development and maintenance environment is an obvious way to exploit CA Enterprise Tester's power and flexibility. Testing is required at many stages of development and implementation cycles and may include the following:

- Unit testing of a single program or module. CA Enterprise Tester scripts may be developed to execute each path in the application code, including all possible error conditions. The Interactive System Record function may be used to record the mainline path through the code. The recorded script may then be edited to check for appropriate expected results, perhaps using a traditional unit test plan as guidance.
- Integration testing at an application or system level. CA Enterprise Tester scripts may be developed which ensure that all paths in an application or system, which require integration points with other applications or systems, are fully executed. People with subject matter expertise may develop scripts in the areas with which they are familiar, and then these scripts may be shared across applications or systems.
- Stress testing at an application or system level. CA Enterprise Tester scripts may be executed concurrently, simulating the behavior of many users on an application, region, or system. This capability makes it feasible to do a level of stress testing that may not have been practical in the past.
- Test data bases may be created from actual production data. CA Enterprise Tester scripts may be developed to select data from production data bases to build test data bases.

- Volume testing of data retrieved from production data bases. CA Enterprise Tester scripts may be developed to extract data from production files or data bases, so that test applications may be executed with actual production data at actual production volumes.
- Regression testing. CA Enterprise Tester provides the capability to capture screen images of an application through the Trace feature, and then use the trace data set as a baseline for a regression test. CA Enterprise Tester Analyzer scripts were developed to automate regression testing.
- Capacity modeling. By exploiting techniques used to do stress and integration testing, the behavior of a processor or complex of processors may be modeled to assist in capacity planning.
- Disaster recovery planning and testing. Although most organizations emphasize planning for disaster recovery, actual testing of the plan may not be feasible. CA Enterprise Tester scripts may be developed which can manage the entire recovery procedure and in many cases, tested on site.

In addition to testing, CA Enterprise Tester scripts may automate a number of tasks associated with development and maintenance implementation:

- CA Enterprise Tester scripts may automate change control procedures, including managing test and production libraries, and ensuring synchronization of source and load versions of the code.
- CA Enterprise Tester scripts may be used to convert files and data bases.
- CA Enterprise Tester scripts may be used to manage batch jobs which, in turn, manage data bases or files.
- CA Enterprise Tester scripts may be used to manage installation of releases or upgrades to existing systems, including verification procedures.
- Using techniques developed for stress and integration testing, CA Enterprise Tester scripts may be used to tune performance of data bases and files.
- The remaining sections of this chapter include sections on regression testing and multiple session tests, and include sample scripts which may be used as a guide to developing installation-specific scripts. The scripts may have environmental dependencies which must be modified before they can successfully execute in any specific system. For example, log on or sign on procedures vary greatly from system to system.

For additional samples of scripts which perform a variety of utility functions, please refer to the appendix “Sample CA Enterprise Tester Scripts”.

Regression Testing

CA Enterprise Tester is designed to automate tasks performed by people using terminals. Any task that is repetitive in nature, tedious, time-consuming, error-prone, or data-input intensive is a candidate for CA Enterprise Tester automation. Regression testing is a task that may be described by all of the previous phrases and a perfect candidate for CA Enterprise Tester automation.

Regression testing refers to procedures designed to verify that unchanged components of a system continue to perform as designed, and to identify and prevent the introduction of unplanned changes or errors into production systems.

Since regression testing should focus on all of the components of a system which should not change, comprehensive testing of this type is often not done because of cost and time constraints, and the tedious, repetitive and error-prone nature of the task.

CA Enterprise Tester sessions, session traces, and session analyzers may be used to automate regression testing.

CA Enterprise Tester sessions may be used interactively in the same way that real terminal sessions are used to access online systems and enter transactions, to test each function performed by an application.

CA Enterprise Tester sessions that are used interactively to test application software may be traced to capture each terminal input entered by a user and each resulting transaction response in a trace data set.

CA Enterprise Tester scripts may also start and use virtual terminal sessions that look like ordinary terminal sessions to an online system, and may read and process screen images that were previously captured in a trace data set when an interactive user session was traced:

- Using the PSL TRACK statement, a script may read an entire screen image and other information about a screen image and session into a TRACKED virtual terminal display buffer and PSL variable fields.
- Using the PSL LOGON statement, a script may start a new virtual terminal session that is similar to a session used by a person when an interactive user session was traced.
- Using the PSL TYPE statement, a script may rekey input data from a TRACKED screen image that was previously entered by a terminal user into a virtual terminal display and may send the input by pressing the Enter key, PF key or other key used to send the original input.
- Using the PSL IF and MASK statements, a script may easily compare new transaction responses to old TRACKED responses to find differences deemed significant by a user, and may use other general PSL statements to analyze and report exactly what differences are found.

CA Enterprise Tester scripts that create and use new virtual terminal sessions to repeat previously traced transactions and that analyze and report differences between new and old results are called session analyzers, and may be used to perform efficient and reliable regression testing automatically whenever online applications or systems are modified.

Using CA Enterprise Tester to automate regression testing eliminates the considerable cost and effort required to perform regression testing manually and improves the quality of online systems by reliably pinpointing all unplanned changes during testing, before any serious harm is done.

The topics that follow describe CA Enterprise Tester functions and applications that are used to automate regression testing, and explain how to use them.

Application Tests and Test Data

Application tests that are to be repeated by CA Enterprise Tester to perform regression testing automatically must first be performed by a person using a CA Enterprise Tester session and session trace to capture the tests and test results in a trace data set. The trace FORMAT option must be specified when a traced session is to be used to perform regression testing. Starting Virtual Terminal Sessions, Using CA Enterprise Tester Sessions and Tracing Session Screens, provide complete information that explains how to start and use CA Enterprise Tester sessions and session traces.

CA Enterprise Tester may be used by individuals to perform regression testing while developing or maintaining software to improve personal productivity or may be used as a tool to implement formal quality assurance standards and procedures. When CA Enterprise Tester is to be used as a tool to implement formal quality assurance standards and procedures, consideration should be given to how application tests and test data are captured, maintained and used to minimize required test data maintenance, and to maximize the flexibility and reliability of a formal regression testing plan. For example:

- If online data is changed by application tests and the change would cause the application to respond differently if the traced test were repeated, then consideration should be given to backing up application data before a session is traced, and restoring the data before repeating the application tests.
- In many cases, repeatable traces may be created without the need to backup and restore data, by planning test sessions that include transactions that create the data to be used for subsequent tests and that delete the data after all tests have completed.

- While it is very easy to capture complete sessions in a single trace data set that include many application tests or tests of many applications, maintenance of traced application tests may be reduced by capturing fewer application tests and tests of different applications as separate library members. These individual members, then, may be replaced by tracing only a few manual tests when significant changes to particular applications render previously traced application tests obsolete.
- Traced application tests that include logon and sign on activity restrict possible uses of the test data. For example, a trace that starts a session with a particular system can not be used to repeat the traced application tests in a different system. A trace that enters a user ID and password in most cases can not be shared with or used by other users. CA Enterprise Tester provides mechanisms to isolate system access activity so that each future test may be directed to any appropriate system, by any authorized user of that system.

Because session recordings and session traces can be started and ended at any time while a CA Enterprise Tester session is in progress, it is relatively easy to record a script to start a session and to sign on, and to capture application tests in multiple trace library members. Session analyzer options that allow scripts and traces to be put together to perform multiple tests using a single session are explained in the following sections.

The CA Enterprise Tester Session Analyzer

The basic building blocks of CA Enterprise Tester's flexible regression testing capabilities are the special purpose PSL statements and functions that make it easy to develop high-level session analyzers that use and analyze captured session data and create and use new sessions to perform regression testing efficiently, reliably and automatically.

While the flexibility exists and the task is easy, it is not necessary to develop session analyzers to use CA Enterprise Tester to automate regression testing. A distributed general purpose session analyzer, PTESA, performs functions and supports options that may be used to customize processing to perform functions preferred by the user or required by applications being tested.

- PTESA can be used to perform regression testing interactively or to perform regression testing automatically in an unattended batch mode.
- PTESA reports test results in a comprehensive and uniform manner regardless of the environment used to perform regression testing.
- PTESA supports synchronization options that repeat tests reliably even when responses are queued or delayed unpredictably or sent piecemeal.
- Application tests may be captured as session segments to minimize test data maintenance and may be selectively repeated in a single run.

- CA Enterprise Tester scripts may be used to automatically mask predictable differences like date or time to suppress unwanted response mismatches.
- CA Enterprise Tester scripts may logon and sign on for each user to allow userid-free, system-id-free tests to be repeated by any user in any system.
- CA Enterprise Tester scripts may be used to supplement or modify data rekeyed from old application tests to compensate for moved or added application data areas.
- CA Enterprise Tester scripts may be used to customize processing to process predictable delayed responses that may be received at unpredictable times.
- Traced application tests data may be upgraded automatically when tests are repeated by tracing new sessions that reflect application changes.

Analyzing Traced Sessions

PTESA may be started using the Analyze Panel or the Play Panel to perform regression testing interactively, may be started using batch JCL to perform regression testing in an unattended mode, or may be linked to by scripts running in either environment to perform regression testing specified by a script.

PTESA processes either a single CA Enterprise Tester trace data set that was previously capture using the trace FORMAT option, or processes a list of FORMAT trace data sets named in a specified trace list data set. Depending on how PTESA is started, the trace data set or trace list data set to be processed may be specified using panel data entry areas, script parameters, script variables, or a JCL DD statement.

- When PTESA is started using the Analyze Panel, the trace or trace list data set to be processed is specified using Analyze Panel fields:

  Specify trace data set or trace list data set to be analyzed:

  ```
  Data Set Name   ===> 'UID1.PROTERM.TRACE'
  Member Name  ===> MEMBER1  (Blank or pattern for member selection list)
  (Select codes: S Select, B Browse, E Edit)
  ```

- When PTESA is started using the Play Panel, the trace or trace list data set to be processed may be specified as a Play Panel script parameter:

  Specify optional play options and script parms:

  ```
  EXEC PARM  ===> TRACE=UID1.PROTERM.TRACE(MEMBER2)(ie.,
  LIST,FLOW,variable=value,..)
  ```

- When PTESA is started using batch JCL, the trace or trace list data set to be processed may be specified as a JCL EXEC statement PARM or may be supplied as a TRACE DD statement:

```
//PTESA EXEC PGM=PTEBS,PARM='UID1.PROTERM.TRACE(MEMBER3)'
```

or

```
//TRACE DD DSN=UID1.PROTERM.TRACE(MEMBER4),DISP=SHR
```

- A script that links to PTESA may specify the trace or trace list data set to be processed using the PSL SET statement:

```
SET &TRACE TO 'UID1.PROTERM.TRACE(MEMBER5)'LINK 'PTESA'
```

PTESA uses default or specified analyzer option values explained in Session Analyzer Options to process specified trace data. Depending on how PTESA is started and the data set specified as input to PTESA, analyzer option values may be specified using panel fields, script parameters, PSL SET statements or using trace list control statements.

Using Scripts to Set Analyzer Options

The three analyzer scripts listed on the Analyze Panel, PTESAFE, PTESAVE and PTESAVED illustrate a convenient way to package preferred analyzer options to customize analyzer processing.

PTESAFE links to PTESA after setting analyzer DELAYS and SYNCTIME option values that may be used to analyze sessions that send a predictable number of responses to each transaction entered. Provided that the system responds as expected, the options allow PTESA to analyze traced session data at maximum speed without encountering session synchronization problems.

```
set &delays to 'synctime'set &synctime to '1000'link 'ptesa'
```

PTESAVE links to PTESA after setting analyzer DELAYS and TIME option values that may be used to analyze sessions with systems that may send an unpredictable number of responses when transactions are repeated. The options activate PTESA logic that gives the system up to 10 seconds to send possible delayed responses when the state of the session permits the system to send additional responses.

```
set &delays to 'time'set &time  to '1000'link 'ptesa'
```

User-written analyzer scripts that set option values to activate desired analyzer functions and customize analyzer processing may be used to tailor analyzer processing for an individual user and may be used in any environment to perform regression testing in a consistent manner. For example, the following analyzer script could be used instead of PTESAVE to activate similar synchronization options, but with shorter time delays, to link to a personal logon and logoff script to start and end sessions used to analyze trace data that does not contain logon, signon or logoff transactions, and to automatically capture an upgraded trace for each analyzed trace data set whenever regression testing is performed.

```
set &delays to 'time'set &time  to '0500'set &logon to 'mylogon'set &logoff
to 'mylogoff'set &tracepds to 'my.upgrade.trace'link 'ptesa'
```

Trace Lists

The input to PTESA may be either a single trace data set or may be a trace list that contains a list of trace data sets to be processed and may specify analyzer options to be used to process all or each trace data set listed. A trace list is not a CA Enterprise Tester script, but rather is a data set that contains keyword control statements or comments and includes one or more ANALYZE control statements that each specify a trace data set to be processed by PTESA.

A trace list keyword begins in position one of each trace list record and is followed immediately by an equal sign, '=', and an analyzer option value. Trace list keywords, except for ANALYZE are named the same as analyzer option variables explained in Session Analyzer Options. The following example illustrates the format of a trace list:

```
* This trace list specifies scripts to start sessions and mask
* unwanted mismatches, specifies an upgrade trace library and
* specifies 3 trace library members to be
analyzed.DELAYS=SYNCTIMESYNCTIME=1000LOGON=mylogonLOGOFF=mylogoffTRACEPDS=my.
newtrace.libraryMASK=mymask1ANALYZE=my.trace.library(add)ANALYZE=my.trace.lib
rary(update)MASK=mymask2ANALYZE=my.trace.library(delete)
```

PTESA processes trace list records sequentially. When each trace list keyword option other than ANALYZE is processed, PTESA sets the corresponding analyzer option variable to the value specified by the trace list keyword option. When an ANALYZE keyword statement is processed, the specified trace data set is analyzed using current analyzer option values. As a result, different analyzer options may be used to process each trace data set analyzed.

An advantage in using trace lists is that multiple trace data sets, that each contain a partial session and a few related application tests, may be repeated sequentially using a single session. By structuring data in this way, if an application is modified to the extent that traced transactions are not repeatable, and the application is used in one small trace data set instead of large multi-application trace data sets, it is relatively easy to replace the small obsolete trace data set by tracing a relatively small number of tests of the new version of the software, that are entered manually.

Another advantage in structuring traced application tests in small trace data sets is that application tests may be repeated selectively when only a few tests are required to test minor application changes.

Trace lists used by PTESA may also be used as input to the Browse Panel to view several trace data sets as a single session or may be used as input to the CA Enterprise Tester Trace or Record Utilities to print data sets as a single session or to create scripts that would repeat all traced transactions if executed.

Session Analyzer Options

The distributed session analyzer, PTESA, supports options that may be used to customize analyzer processing or to activate optional analyzer functions.

When they are used by PTESA, analyzer options are PSL variable fields that contain option values. Option values may be entered in panel fields, may be assigned using JCL parameters, may be set directly by scripts that link to PTESA, or may be specified using trace list control statements. While the syntax of the parameters, PSL SET statements or trace list control statements used to assign option values to an analyzer option variables differ, each method of specifying option values refer to the option by the analyzer option variable name and specify an option variable value documented below. Analyzer option variable values may be specified using any alphabetic case. Option values that are documented as constants may be truncated to a single character. For example, 'Yes' may be specified as 'Y', or 'Pause' may be specified as 'P'.

Analyzer Options: Session Synchronization

DELAYS

The analyzer DELAYS option may specify one of the values Synctime, Time, Bracket, or Input to specify a synchronization option to be used by the analyzer to synchronize a live session with traced session data being analyzed.

- Sync time instructs PTESA to use the PSL INVITE statement and the time specified using the SYNCTIME analyzer option only to receive anticipated consecutive outbound messages implied by consecutive outbound messages found in trace data. The option value is appropriate only for sessions that send a predictable number of responses to each transaction entered.
- Time instructs PTESA to use the PSL DELAY statement and a time value specified using the TIME analyzer option to receive possible delayed responses that may follow an outbound message that leaves the analyzer session in the CONTENTION state. Either DELAYS Time or the Bracket option should be used to synchronize the analyzer session with the traced session when testing systems that may respond unpredictably.

- Bracket instructs PTESA to execute PSL DELAY statements found in traced session data to receive possible delayed responses that may follow an outbound message that leaves the analyzer session in the CONTENTION state. If DELAY statements are not found in traced session data, a DELAY is issued that uses the time specified by the TIME analyzer option.
- Input instructs PTESA to execute all PSL DELAY statements found in trace data even if there is no possibility that the online system may send a message or to issue a DELAY statement using the time specified by the TIME analyzer option if DELAY statements are not found in traced session data.

SYNCTIME

The analyzer SYNCTIME option may be used to specify a time expressed in the form, ssth, (seconds, tenths and hundredths of a second), that is to be used as an INVITE time when the DELAYS Sync time option is active and consecutive outbound messages are expected on a live session that correspond to consecutive outbound messages found in a trace data set being analyzed. The default value is 1000 (10 seconds).

TIME

The analyzer TIME option may be used to specify a time expressed in the form, ssth, (seconds, tenths and hundredths of a second), to be used as a PSL DELAY statement time when the analyzer DELAYS Time option is active, or when the DELAYS Bracket or Input option is active and a PSL DELAY statement is not found in a trace data set, to receive possible delayed responses that may be received following a response that leaves the analyzer session in the CONTENTION state.

DELAYPCT

The analyzer DELAYPCT option may be used to specify a percent from 0 to 999 that is a percent of the time specified by PSL DELAY statement found in trace data sets, when the DELAY statements are executed by PTESA while either the DELAYS Bracket or the Input analyzer option is active.

Analyzer Options: Data Comparison

IMAGE

Normally, PTESA compares all corresponding data contained in trace and live session screens, including data contained in non-display screen fields, as binary values. IMAGE may be set to Yes to request that the image of screen data be compared. The image of screen field data is what a person sees on a screen. When IMAGE is set to Yes, data in non-display screen fields and binary zeros are considered to be blanks when screen data is compared.

CASE

By default, PTESA is sensitive to differences in the case of alphabetic characters that are compared. CASE may be set to No to specify that corresponding upper and lower case alphabetic character values are to be considered equal when they are compared.

ATTR

By default, PTESA detects and documents any differences in corresponding screen field attributes. ATTR may be set to No to specify that field attributes at corresponding trace and live session screen locations are to be considered equal.

Analyzer Options: Script Customization

LOGON

The analyzer LOGON option may be used to name a script that is to be linked to before analyzing traced screens. The LOGON script should start a session to be used to analyze traced screens and should enter sign on or other transactions needed to position the session at the screen that matches the first traced screen to be analyzed.

LOGON scripts allow traced session data to be free of user ids and passwords and allow traced materials to be shared. To create a personal LOGON script, start a recording, use the SESSION command to start a session, sign on, enter any transactions needed to get to a primary menu, then end the recording. User-written LOGON scripts, containing logic to handle variable session start up events or that get a current password from an external source, may also be used.

MASK

The analyzer MASK option may be used to name a script that is to be linked to before comparing each pair of subsequently analyzed trace and live session screens.

A MASK script may examine the screens and execute Script Language MASK statements to suppress the comparison of insignificant data like date and time displays. For example, the following script masks the time displayed on the ISPF/PDF PRIMARY OPTION MENU, at row 4, column 73.

```
MASK if tracked <1 1 80> contains 'ispf/pdf primary option menu' then  mask
<4 73> 'nn:nn'  endif
```

Masking Dates, Times, and Other Data and MASK describe CA Enterprise Tester masking capabilities and provide detailed descriptions of MASK statement syntax and options.

TYPE

The analyzer TYPE option may be used to name a script that is to be linked to after TRACKED data has been rekeyed but prior to the AID key being typed. A TYPE script may modify or supplement data rekeyed from an old traced screen image when application input fields are moved or added. Upon return from the script, the AID key from the traced session is typed.

EDIT

The analyzer EDIT option may be used to specify the name of a script that is to be linked to by PTESA prior to comparing a TRACKED screen image to the current virtual terminal display.

An EDIT script may manipulate the trace dataset and/or the live session if the script determines that a correctable out-of-sync condition exists.

A sample EDIT script, PTETSOED may be used with TSO sessions to keep traced and live sessions in sync regardless of the number or order of TSO stars, (***), messages received when a trace was captured or when a traced session is analyzed.

FILTER

The analyzer FILTER option may be used to name a script that is to be linked to after starting a session and when the terminal keyboard is freed following each re-keyed terminal input. The script may examine the traced input screen, the live session screen, and may issue INVITE or TYPE statements to manipulate the live session in any desired way.

SYNC

The analyzer SYNC option may be used to name a script that is to be linked to after processing a trace data sets named by a trace list ANALYZE statement.

After a trace data set is processed, a previously specified SYNC script may verify that a session has returned to a primary menu, that is the assumed starting point of session activity contained in subsequently processed trace data sets. If an intended menu is not displayed, a SYNC script may enter transactions needed to return to the menu, allowing subsequent trace data sets to process correctly. For example:

```
SYNC if <1 1 80> contains 'ispf/pdf primary option menu' then  next else
type home 'return' erase enter  endif
```

LOGOFF

The analyzer LOGOFF option may be used to name a script that is to be linked to after processing all trace data being analyzed. A LOGOFF script may execute any transactions needed to end a session in an orderly manner.

Analyzer Options: Upgrading Traces

TRACEPDS

The analyzer TRACEPDS option specifies the data set name of a trace library to be used to capture new sessions created by repeating transactions found in the old trace data being analyzed. When a TRACEPDS is specified, a new member is created in the specified TRACEPDS that is named the same as each old trace member analyzed. Each new TRACEPDS member is opened and closed at the same time as the input trace member to create upgraded trace members that contain session events equivalent to the events in the old trace members.

SCREENS

The analyzer SCREENS option may be used to specify what screens are to be captured in an upgraded trace members that are created when the TRACEPDS analyzer option is active. Freekb specifies that outbound screen images are to be traced only when messages are received that free the terminal keyboard. Chains specifies that outbound screen images are to traced when any complete outbound message is received.

Analyzer Options: Event Handling

MISMATCH

The analyzer MISMATCH option may be used to specify the action to be taken when response mismatches are detected by the session analyzer. Valid actions are:

- Continue instructs PTESA to continue processing without interruption after documenting each response mismatch.
- Pause instructs PTESA to stop and wait for terminal user action after documenting each response mismatch. Because PTESA cannot communicate with a terminal user when it is running in batch, Pause is changed to Continue when PTESA is used to perform regression testing in batch.
- Stop instructs PTESA to stop processing after documenting a response mismatch. When PTESA is used to perform regression testing interactively, Stop is changed to Pause allowing the terminal user to decide whether to continue or end processing.

The default MISMATCH action is Pause.

KEYERROR

The analyzer KEYERROR option may be used to specify the action to be taken when the analyzer can not rekey trace input data into the live session screen. Valid actions are:

- Continue instructs PTESA to document the problem but to continue processing by entering the transaction even though all input data was not rekeyed.
- Pause instructs PTESA to stop and wait for terminal user action after documenting the problem. Because PTESA cannot communicate with a terminal user when it is running in batch, Pause is changed to Continue when PTESA is used to perform regression testing in batch. When PTESA is used interactively and pauses because it is unable to rekey traced input data, the transaction is not entered. The terminal user may select the live session, and may key in any data and enter the transaction manually. After completing a transaction manually, PTESA may be resumed to repeat the next traced transaction.
- Stop instructs PTESA to stop processing after documenting the problem. When PTESA is used to perform regression testing interactively, Stop is changed to Pause allowing the terminal user to complete the transaction manually and to decide whether to continue or end processing.

The default KEYERROR action is Stop.

Analyzing Sessions Interactively

The Interactive System Analyze Panel, which is explained in detail in Analyzing Traced Sessions, may be used to start a session analyzer to repeat and analyze traced CA Enterprise Tester sessions that were captured using the trace Format option, as described in Tracing Session Screens.

When the Analyze Panel is used to analyze traced session data, the trace data to be processed by the session analyzer is entered at the Analyze Panel Data Set Name field. The data set specified may be either a single trace data set or a trace list data set containing trace list statements that specify analyzer options and one or more ANALYZE statements that specify trace data sets to be processed.

If the analyzer script, trace list or Analyze Panel specify scripts to be used by the session analyzer, that are in a private script library, then the name of the library must be entered at the Analyze Panel.

A mask script may be specified that uses the PSL MASK statement to suppress unwanted response mismatches like date or time.

The analyzer script to be used, PTESAFE, PTESAVE, PTESAVED or a user-supplied script that sets preferred analyzer options and links to PTESA to analyze specified session data, must be entered at the Analyze Panel.

```
+-------------------------------------------------------------------------------+
| ProTerm Analyze Panel ------------------------------------------ Port 001     |
| COMMAND ===>                                                                  |
|                                                                               |
|      ENTER TO VIEW I/O    PF15 END ANALYZE REQUEST  K  KEYS PANEL             |
| PF23 STEP TO SCREEN 1    LOG LOG PANEL   O  OPTIONS PANEL                     |
|  Specify trace data set or trace list data set to be analyzed:                |
|                                                                               |
| Data Set Name  ===> 'UID1.PROTERM.TRACE'                                      |
| Member Name    ===> TESTAP ( Blank or pattern for member selection list )     |
|                            ( Select codes: S Select, B Browse, E Edit )       |
| Specify Script Library to access private session analyzer scripts: **         |
|                                                                               |
| Script Library  ===> 'UID1.PROTERM.SCRIPT'                                    |
| Mask Script     ===> MASKAP  ( * or pattern for member selection list )       |
|                                                                               |
|                                                                               |
| Analyzer Script ===> PTESAFE ( PTESAFE  Fast Fixed Event Session Analyzer)    |
|                              ( PTESAVE  Timed Variable Event Analyzer )       |
|                              ( PTESAVED Delayed Variable Event Analyzer )     |
|                              ( Blank or pattern for member selection list)     |
|                                                                               |
| ** Script Library is searched before SYSTEM.PROTERM.SCRIPT                    |
+-------------------------------------------------------------------------------+
```

A specified analyzer script is started when a valid trace data set or trace list is entered at the Analyze Panel using the enter key, VIEW command or STEP command. If the enter key or VIEW command is used to start an analyzer script, the new session created by the analyzer is displayed as a session movie that proceeds until a mismatch or other problem is detected. The STEP command or PF key may be used to start an analyzer script and may be used to advance analyzer processing to the next inbound session step or system response, one screen at a time, until a mismatch or other problem is detected by the session analyzer. The Monitor Panel is displayed when a mismatch or other problem is detected. Messages displayed in the Monitor Panel log explain the mismatch or other problem detected, and state options available to the user, that are explained in following section.

Analyzing Session Mismatches

When PTESA finds a pair of screens that do not match, messages are logged that describe the mismatch and show the data that does not match. When session analyzer MISMATCH Pause option is active, PTESA pauses after documenting a mismatch and waits for user action. When a session analyzer stops, the Monitor Panel is displayed. The Monitor Panel contains a Message Log that may be used to view logged messages. Figure 8-2 illustrates a mismatch message displayed via the Monitor panel Message Log.

```
+-------------------------------------------------------------------------------+
| ProTerm Monitor - Session(PT02/TSOA062) ------------------------  Port 001 |
| COMMAND ===>                                                                  |
|                                                                 SCROLL ===> CSR |
| PF24 VIEW UNTIL STOP/END RES RESUME /P/A/R/T   PF3  END /P/A/R/T/L            |
| PF23 STEP TO I/O OR STOP S SELECT SESSION    LOG  LOG PANEL                   |
| PF22 REVIEW /S/T  EX EXECUTE 'STATEMENT'  K   KEYS PANEL                      |
| Script Status ----------------- 2296 Statements executed; Session I/O 3/12 |
| SCRIPT IS STOPPED                                                             |
| Next Script Line ---------------------------------------- Line number 1047 |
|    endif                                                                      |
| Message Log ------------------------------------------- Log line 150 of 157 |
| Row 3 Display+----+----1----+----2----+----3----+----4----+----5----+----6 |
|    differences |       DD DD  D                                               |
|    old display | RS6 LOGON IN PROGRESS AT 04:13:04 ON DECEMBER 3, 1996       |
|    new display | RS6 LOGON IN PROGRESS AT 16:07:07 ON DECEMBER 3, 1996       |
| REVIEW, then ROLL to visually compare full screen displays.                   |
| STEP or VIEW to repeat the next traced transaction.                           |
| PTE10150 - SCRIPT PAUSE, AWAITING INTERACTIVE SYSTEM COMMAND                  |
| Stop Conditions ---------------------------------- ( Enter stops or all )  |
| Script Verbs ===>                                                             |
| Statement Labels  ===>                                                        |
| Statement Numbers ===>                                                        |
+-------------------------------------------------------------------------------+
```

The Monitor Panel has many functions that are described in other sections of this Application Guide. Some of the information and options provided by the Monitor Panel require an advanced understanding of CA Enterprise Tester scripts. To get started, begin by focusing on the Monitor Panel Message Log. The Message Log is a scrollable running log of information that is used by a session analyzer to present information about mismatches detected while analyzing a session. The Monitor Panel is displayed when an interactive session analyzer detects a mismatch and stops to wait for user action.

Each message identifies the total number of screens the analyzer has compared and the row of the current screen which contained a mismatch. The trace data set name, trace sequence number within the data set and the time stamp when the trace data was written to the data set are also provided.

When a difference between trace and session data is found, a message is logged which easily identifies the mismatches by placing an D over the trace and session data. All mismatches on the row are identified in this manner. If differences between attributes or non-display data are found, the differences are flagged using the characters A for attributes or N for non-display data.

If screen attributes or mismatches are found in non-display data, then messages are logged that describe the properties of the corresponding attributes or that show non-display data in character and hexadecimal format.

A message is logged if the cursor is returned at a different location.

When a screen has several differences, multiple messages are logged. Scroll UP and DOWN commands, or the PF keys to which the commands have been assigned, may be used to view previous and subsequent mismatch messages.

The difference found in the previous example is an obvious date/time stamp mismatch. The mismatch occurs because the traced session contains the date and time sent by the online software when the trace was created. Quite naturally, when the traced session is repeated, the online software sends the current date and time. Masking Dates, Times, and Other Data describes how to mask these insignificant mismatches.

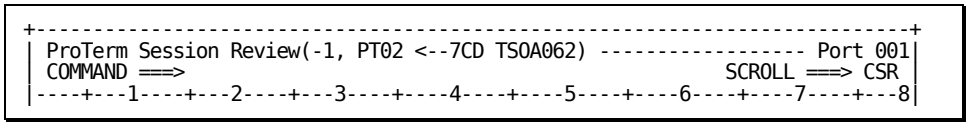
CA Enterprise Tester provides a number of commands, such as REVIEW, ROLL and STEP which may be used to view session and trace screen images to analyze what has caused a mismatch. Command entry may be greatly simplified by assigning commands to PF keys. Refer to PF Keys and Commands for a complete description on how to customize your PF keys.

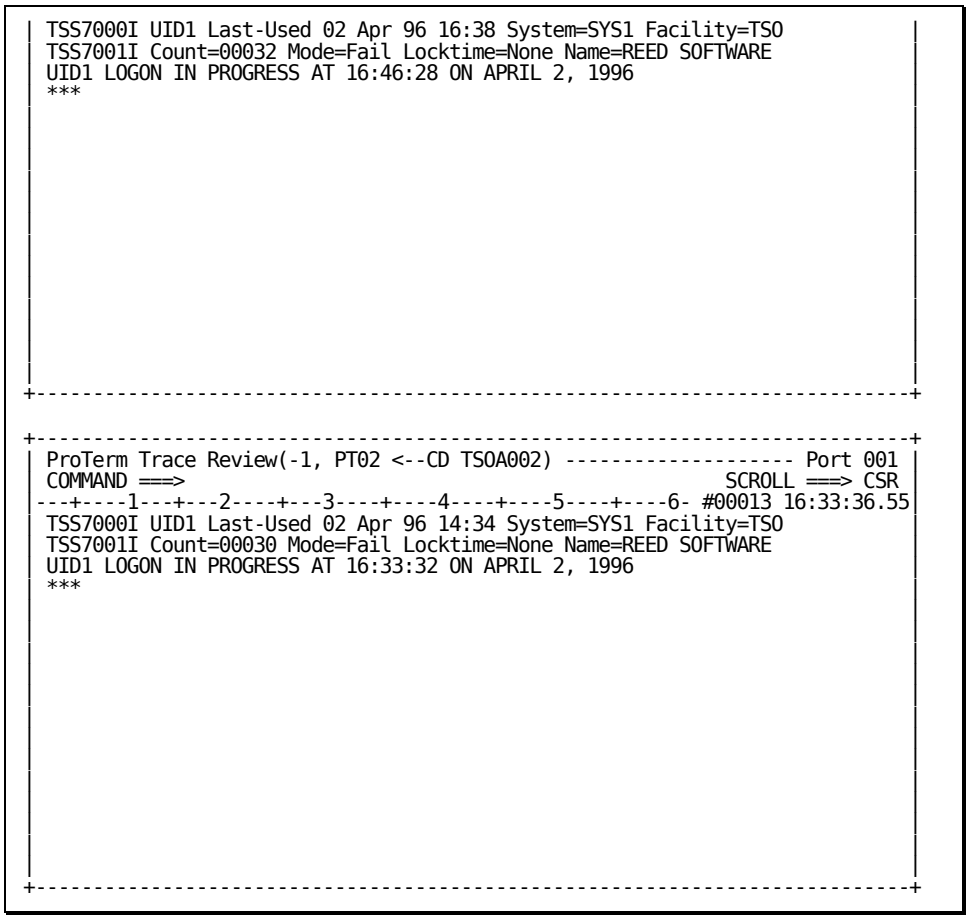
While a session analyzer is stopped at a mismatch, the full live session screen may be viewed by entering the REVIEW command, (or the PF key to which you have assigned the command). In fact the REVIEW command key may be used to back up several session steps to see the screens that preceded the mismatch.

While reviewing session screens, the ROLL command key may be pressed to toggle to the corresponding trace data set review screen. The ROLL command key may be pressed several times to toggle between corresponding session and trace screens. Because the ROLL command key changes only the data that does not match, toggling between trace and session review screens causes mismatching data to blink, making it easy to see all screen mismatches.

While viewing either session or trace review screens, the session presentation may be switched from Window Mode to Zoom Mode by pressing the PF key to which the ZOOM command has been assigned, and/or a PA2 key, (depending on how the PA2 switch option has been set). ROLL, REVIEW and STEP command keys work in either mode provided that both Window Mode and Zoom Mode PF keys have been assigned to the commands.

The following example illustrates the presentation of session and trace review screens in Window Mode when the REVIEW and ROLL command keys are pressed.





Additional information about the CA Enterprise Tester session review feature may be found in Interactive Session Review.

While viewing session or trace review screens, you may return to the Monitor Panel by pressing the END command key.

At any time, while reviewing session or trace screens or while viewing the Monitor Panel, you may continue the session analysis by entering a VIEW, WAIT or STEP command or key, to proceed to subsequent session screens. If a STEP command is entered while reviewing other than the most recent review screen, more recent review screens are displayed.

If you are analyzing a session, and one or more screen fields have been moved to a new location on a screen, you may receive a message that describes a re-keying error. A re-keying error occurs when a session analyzer attempts to enter keystrokes obtained from a trace screen image, into a field on the live session screen that has been changed to a protected field. When this happens, you may continue the session by manually re-keying the screen data. To manually key data at the new correct location, first press the STEP command key, to ignore the re-keying error message and to advance the session to the point where the next input screen is about to be sent to the online system. When viewed using Window Mode, the panel identifies the display as a Session Step, and the arrow, -->, pointing at the name of the online system identifies the step as an inbound session step.

While viewing the inbound session step, you may change the input data as desired, and may even change the AID key to be used by pressing a PF key that has been assigned to a CA Enterprise Tester PF key command (see PF Keys and Commands), or by pressing the Enter key. Entering data or pressing the PF key or Enter key does not resume the session analysis, but rather just updates the data to be sent or the AID key to be sent when the analysis is resumed. After manually keying data, the session analysis may be continued by pressing the VIEW, WAIT or STEP command keys. The manually keyed data will be sent instead of or in addition to the data that was automatically re-keyed from the trace data set.

If a session trace is obsolete, because a few fields have been moved on application screens, or if fields have been added and additional input data must now be entered, then this technique may be used to update an old session trace. The TRACE command may be used to start a session trace to capture a part of or an entire session that is re-created by a session analyzer. When keystrokes are manually entered at a session step, the new trace will include both script entered keystrokes and manually entered keystrokes. By stepping through a session and manually keying new or changed fields, the resulting trace will contain the input now required by the modified online system software being tested.

While analyzing a session, if session mismatches are detected that make it useless to continue the analysis, an END ANALYZE command, abbreviated e a, may be entered to terminate a session analyzer. If a session analysis is ended prematurely, you may need to end the session manually. This may be accomplished by entering an END SESSION command or by entering the logoff transaction that is supported by the online system being tested.

Script Monitoring and Interactive Debug provides a general description of more advanced capabilities that are available while using the CA Enterprise Tester Monitor. Advanced Analysis Techniques describes how other Monitor Panel options may be used to gain more information about a mismatch or to continue a session when traced and active sessions get out of sync.

Masking Dates, Times, and Other Data

Online systems often display the current date and time. When a session is traced, traced screen images capture the date and time at the time the trace was created. When a session is analyzed, a new session is created and traced data is entered to automatically re-create the session. When an online application responds to an input and the response includes a current date or time display, then a mismatch will be detected when the trace and session date or time is compared.

This section describes CA Enterprise Tester features that may be used to mask the comparison of dates, times and other insignificant data, allowing a session analyzer to proceed until important session differences are detected.

The Analyze Panel provides fields that may be used to identify a CA Enterprise Tester script that makes decisions about what fields on particular screens are to be considered insignificant. When a trace list is analyzed, mask scripts may also be used to specify and to change mask statements while a session is in progress. Refer to Session Analyzer Options for information about trace list MASK statements.

When a Script Library and mask script name is specified, the script is executed just before comparing a pair of screens. Using appropriate Script Language statements, the script may examine the session and trace screens that are about to be compared and may execute Script Language MASK statements that cause specified screen fields to be considered equal. A single mask script may be developed to make decisions about any number of application screen images. Because masking decisions are external to traced screen images, a single mask script may be used to make decisions for any number of traced sessions without the need to mark up trace data every time a session is traced.

The following example is a complete script that makes decisions to mask particular fields on two selected application screens.

```
if tracked <1 2 *> = 'CESN'
   then mask <1-24>
 exitendifif tracked <1 1 80> contains 'accounts payable menu'
   then mask 'zn-nn-nn' <1 67>
 mask 'zn:nn' <1 76>
 exitendif
```

The first IF statement examines screen row 1, column 2, contained in the current trace screen image, and if that location has the value CESN, specifies that screen rows 1 through 24 are to be considered equal data when the trace screen is compared to the live session screen. The EXIT statement ends execution of the mask script.

If the current traced screen is not the CESN screen, then the next IF statement is executed.

The second IF checks to see if row 1 in the trace screen contains the Accounts Payable Menu title. If it does, then two MASK statements are executed that specify that if the data at screen row 1, column 67, looks like a date, then the date should not be compared, and that if the data at screen row 1, column 76 looks like a time, then the time should not be compared.

If neither condition is matched, then the full trace and session screens will be compared.

When relationships are compared using relational mnemonics such as EQ or relational symbols such as an equal sign, =, CA Enterprise Tester is sensitive to the case of literal data. When you compare a screen location to a literal, be careful to use the case displayed on the application screen. The CONTAINS condition conducts a case insensitive, attribute insensitive search of a source field for a specified argument. CONTAINS allows key screen values to be found without coding specific screen locations.

Complete documentation of the Script Language IF statement may be found in IF.

Complete documentation of the Script Language MASK statement may be found in MASK.

While developing a mask script, you may need to know the location of screen fields that are to be used as the basis for masking decisions and the location of any fields that are to be masked. There are two easy ways to obtain this information without counting rows or columns while using terminal cursor positioning keys.

1. If you are using a terminal that shows the cursor position on the status line, establish a full-screen (Zoom Mode) session with the application and simply position the cursor on each field that is the basis for a masking decision or that is a field to be masked.
2. The CA Enterprise Tester SPOT command may be used to display the screen location of the cursor. First, use the Keys Panel to temporarily assign one of the PF keys to the SPOT command. Then, start a CA Enterprise Tester session with the online system and display each screen that is to be masked using Window Mode. Position the cursor on each field that is the basis for a masking decision or is a screen field to be masked. When the SPOT command key is pressed, the current cursor location, relative to the virtual terminal screen is displayed in the short message area of the Session Panel.

To use a masking script, enter the Script Library and script name in the optional panel fields of the Analyze Panel, or use a trace list and trace list MASK statement(s) to name the script that is to be used to make masking decisions.

Advanced Analysis Techniques

This section describes advanced techniques that may be used while using CA Enterprise Tester session analyzer interactively. The techniques require an understanding of the CA Enterprise Tester Script Language and debugging capabilities that are supported by the CA Enterprise Tester Monitor. The CA Enterprise Tester Script Language is documented in The CA Enterprise Tester Script Language. The CA Enterprise Tester Monitor is described in Script Monitoring and Interactive Debug.

While analyzing a session, when a mismatch occurs that involves mismatching screen attributes or non-display screen data, then the mismatch messages logged on the Monitor Panel Message Log will include a description of mismatching attributes or will show the non-display data in character and hexadecimal formats. The unequal data displayed when a mismatch is encountered is displayed as it appears on a screen. When a field of non-display data mismatches, it is displayed in character and hexadecimal formats. The EXECUTE command can be used to display non-display data in visible or hexadecimal form. For example:

```
COMMAND ==> ex 'log tracked <1,65,8>'
```

or

```
COMMAND ==> ex 'set &hex from hex <1,65,8>  log &hex'
```

The first example logs 8 characters of data that is contained in the current trace screen image at row 1, column 65. The second example converts 8 characters of data contained in the live session screen at row 1, column 65, to external hexadecimal characters, and logs the hexadecimal characters.

Under certain circumstances, traced and active sessions may get out of step. For example:

- If an online system fails to send a message that was appropriate when the trace was created, or if an online system sends an extra message that was not appropriate when the trace was created, then continued re-keying of traced terminal inputs may be illogical or impossible.

When an active session gets out of step with a traced session that is being analyzed, then continuation of the session may be illogical unless the trace or the session can be repositioned. For example:

- Keying errors will occur if a traced screen field is re-keyed into a protected field on the wrong active session screen.
- Meaningless mismatches will occur when the wrong trace screen image is compared to the active session screen.

When these events occur, and after logging appropriate messages, a session analyzer may stop and wait on user action. By reviewing the session and trace screen images being re-keyed or compared, it will usually be obvious that the screen being re-keyed does not correspond to the active session screen or that a mismatch has occurred because the wrong trace and active session screens are being compared.

While the analyzer is stopped at a re-keying error or mismatch, the active session can be selected and used interactively, by entering the SESSION command at the Monitor, Session Review or Trace Review panel.

- If a re-keying error was encountered, the session analyzer is attempting to re-key the most recent Trace Review screen. The active session can be selected and can be interactively manipulated to display the screen that corresponds to the trace screen that is being re-keyed. When the correct screen is displayed, data can be manually keyed into the live session screen, being careful not to press Enter , and the VIEW, WAIT or STEP command key may be used to resume session analysis.
- If a mismatch was detected because the sessions got out of step and the wrong screens are being compared, then the active session can be selected and interactively manipulated to position the session at the screen that corresponds to the most recent Trace Review screen. The VIEW, WAIT or STEP command key may be used to resume the session analysis directly from the session display.

Analyzing Sessions in Batch

The distributed session analyzer, PTESA may be used to perform regression testing interactively or may be played using scheduled or submitted batch jobs to perform unattended regression testing automatically.

Given the same analyzer options, PTESA processes traced session data in the same way when it is running in unattended batch mode as it processes the data when it is used to perform regression testing interactively, with one exception. When it is running in unattended batch mode, PTESA can not stop and wait for user action when a mismatch is detected or it is unable to rekey traced input data, and therefore changes MISMATCH=PAUSE and KEYERROR=PAUSE analyzer options to MISMATCH=CONTINUE and KEYERROR=CONTINUE.

When PTESA is used to perform unattended regression testing in batch, the traced session data to be processed and analyzer options are specified using JCL instead of panel fields.

- The name of the session analyzer to be executed, PTESA in this case, is specified on the PTEIN DD statement. Session analyzer messages are written to the PTELOGDD statement. A mask script is specified using the JCL PARM field. Some trace list keywords specify a script name which will be invoked by the CA Enterprise Tester LINK statement. The named script must be a member in the libraries specified by the PTELIB or PTEIN DDs or the system script library which was defined at the time of installation. The order in which the concatenated libraries are searched for the script is described in LINK.

  The input to the session analyzer is a single session trace data set that contains a complete session. The data set name is specified on the TRACE DD statement. Because MISMATCH and KEYERROR parameters are not specified, the session analyzer will continue processing after documenting mismatches or will stop if data can not be re-keyed.

  ```
  //ATRACE EXEC PGM=PTEBS,PARM='MASK=MASK1'
  //PTEIN   DD DISP=SHR,DSN=ProTerm.Script.Library(PTESA)
  //PTELOG  DD SYSOUT=A
  //PTELIB  DD DISP=SHR,DSN=mask.script.library.name
  //TRACE   DD DSN=a.trace.dataset.name,DISP=SHR
  ```

- In this example, a trace list is analyzed that is provided as in-line TRACE DD data. A PTETRACE DD statement is used to capture the complete re-created session. All scripts that are named by LOGON, MASK, and LOGOFF Trace List statements are contained in the PTELIB or PTEIN DDs or implicitly in the system script library. The LOGON and LOGOFF scripts start and end the session that is used to analyze the partial session trace data sets named by ANALYZE statements. Multiple MASK statements are used to specify different masking scripts to be used during the analyzing subsequent trace data sets specified by ANALYZE statements.

  The JCL PARM field is used to specify two additional analyzer options. The TIME analyzer option is specified to delay 3 seconds when a system response causes the session to enter the CONTENTION state, to give the system time to send and delayed response that would cause sessions to get out-of-sync if a delayed response were not received. The KEYERROR parameter specifies that processing should continue even when some trace data can not be re-keyed.

  ```
  //ALIST EXEC PGM=PTEBS,PARM='TIME=300,KEYERROR=C'
  //PTEIN   DD DISP=SHR,DSN=ProTerm.Script.Library(PTESA)
  //PTELIB DD DISP=SHR,DSN=script.library.name
  //PTETRACE DD SYSOUT=*
  //PTELOG DD SYSOUT=*
  //TRACE  DD *
  LOGON=imslogon
  MASK=mask1
  ANALYZE=ims.trace.lib(imst0001)
  MASK=mask2
  ANALYZE=ims.trace.lib(imst0002)
  ```

```
ANALYZE=ims.trace.lib(imst0003)
LOGOFF=imslogof/*
```

- In this example, the TRACE keyword is used to identify a trace list that is a member of a partitioned data set. The trace list contains trace data set names to be analyzed and other analyzer options. The session analyzer log and a trace of the re-created session are written to new data sets specified on the PTELOG and PTETRACE data sets.

```
//ALIST EXEC PGM=PTEBS,PARM='TRACE=TRACE.LIST.LIB(ALIST)'
//PTEIN   DD DISP=SHR,DSN=ProTerm.Script.Library(PTESA)
//PTELIB DD DISP=SHR,DSN=script.library.name
//PTELOG DD DSN=analyzer.log,DISP=(NEW,CATLG),
// DCB=(DSORG=PS,RECFM=FBA,LRECL=151,BLKSIZE=6040),
// UNIT=DISK,SPACE=(CYL,(1,1),RLSE)
//PTETRACE DD DSN=session.trace,DISP=(NEW,CATLG),
// DCB=(DSORG=PS,RECFM=FBA,LRECL=135,BLKSIZE=6075),
// UNIT=DISK,SPACE=(CYL,(1,1),RLSE)
```

Multiple Session Testing

Using CA Enterprise Tester session traces, session analyzers, masking scripts, and other features and techniques described in Regression Testing, captured sessions may be used to automate thorough regression testing of online application software whenever it is modified.

CA Enterprise Tester's multiple session and session scripting capabilities may also be used to perform multiple session tests that can be used to evaluate other aspects of online system operation and performance.

- CA Enterprise Tester may be used to perform integration tests that concurrently execute related components of an online application system. Integration testing may be used to detect bottlenecks, deadlocks and other problems prior to installing a new application system into a production environment.
- CA Enterprise Tester may be used to perform network simulations that create controlled demands of an online system. Network simulations may be used to measure the effects of planned increases to online system workload or to measure the effects of online system tuning efforts.
- CA Enterprise Tester may be used to develop stress tests that create online system workloads that exceed the capacity of any system. Load tests may be used to determine an online system's limitations, to measure the capacity of a machine or to compare the relative capacity of different machines.
- CA Enterprise Tester may be used to measure the system overhead of general purpose products like session managers and data stream optimizers while they are being evaluated during a product trial.

The common thread in all of the kinds of tests described above is that multiple concurrent sessions are required. Regardless of the objective of any particular multiple session test, CA Enterprise Tester may be used to create a desired multiple session workload by playing multiple scripts on multiple Interactive System ports concurrently.

While multiple session tests may use session analyzers and traced sessions to drive multi-session online system tests, recorded or user-written scripts that create sessions directly provide greater flexibility and performance when used to perform multi-session testing and are the subject of the following topics, that describe CA Enterprise Tester features and suggest techniques that may be used to prepare and conduct multiple session tests.

A Basic Multiple Session Test

Using the CA Enterprise Tester Interactive System session recording facility, a CA Enterprise Tester session may be captured in the form of an executable CA Enterprise Tester script, as described in Recording Session Scripts. The CA Enterprise Tester Record Utility, may also be used to record scripts that repeat traced sessions that were captured to perform regression testing.

A recorded script may be executed using the Interactive System PLAY command and Play Panel, as described in Playing Session Scripts. When a recorded script is executed, a new session is created, and captured keystrokes are re-entered automatically, to re-create the captured session. To an online system, the re-keyed terminal input looks just like the terminal input that was entered manually.

A single played script runs on a particular Interactive System port. Multiple session online system tests may be performed by playing multiple scripts on multiple ports at the same time. To an online system, scripts that create sessions and run at the same time appear to be different people, who are working at different terminals.

An example will show one way start a basic multi-session test using only recorded scripts and Interactive System commands. Assume that you have recorded three scripts named SESSION1, SESSION2, and SESSION3, and that the scripts are stored in a PDS named USERID.SCRIPT.LIBRARY. The following procedure may be used to play the all three scripts at the same time using Interactive System commands:

1. While viewing the Control Panel, enter the following command:

```
COMMAND ===> p ; m 'userid.script.library' session1 ; e ; retrieve
```

The command is actually 4 commands, separated by the ISPF command delimiter ;. (You can specify a different command delimiter using ISPF Option 0.)

- p displays the Play Panel on the next available port.
- m 'userid.script.library'session1 enters the Play Panel MONITOR command and the script library and script to be played. The MONITOR command instructs CA Enterprise Tester to load the script, but to defer execution and display the Monitor Panel after the script is loaded.
- e is an abbreviation for the END command. When the END command is entered at the Monitor Panel, the played script is left in a stopped state and the Control Panel is re-displayed.
- retrieve is an ISPF command that causes previously entered command input to be re-displayed.

When the commands are entered, the Control Panel is re-displayed, the entire list of commands remain in the command field, and the word STOPPED appears in the script status column for the port, indicating that a script is loaded on the port but that the script is not executing.

2. Play the second script on the next available port by changing the script name in the retrieved command before pressing enter.

   ```
   COMMAND ===> p ; m 'userid.script.library' session2 ; e ; retrieve
   ```

   Again, the command is re-displayed, and STOPPED appears in the script status column of the Control Panel for the port.

3. Play the third script on the next available port by changing the script name in the retrieved command before pressing enter.

   ```
   COMMAND ===> p ; m 'userid.script.library' session3 ; e ; retrieve
   ```

4. When all scripts to be played have been loaded, erase the retrieved command and enter the RESUME command to begin the execution of all loaded scripts at the same time.

   ```
   COMMAND ===> res
   ```

When the RESUME command is entered, all scripts begin executing.

The Control Panel may be used to determine the progress of the test. The information and statistics displayed on the Control Panel are refreshed when Enter is pressed. As each script starts a session, the name of the online system and virtual terminal used for the session appear in Control Panel System and Terminal columns.

The Session column shows the current state of the session and the sum of entered messages and online system responses. IN means that a terminal input is in progress, OUT means that a script is waiting on an online system response, and I/O means that the session is between transactions.

The Control Panel Script column shows the number of script statements that have been executed as illustrated in the following figure.

```
+------------------------------------------------------------------------------+
|ProTerm Control Panel ------------------------------------ 003 Active Ports|
|COMMAND ===>                                                                  |
|                                                             SCROLL ===> CSR  |
| S    SESSION START/SEL   R    RECORD A SCRIPT  STO   STOP /P/A/R/T/#        |
| P    PLAY A SCRIPT       T    TRACE SCREENS    EX    EXECUTE 'STATEMENT'    |
| A    ANALYZE A SESSION   LOG  LOG PANEL PF15   END   (E P, E 1, ETC)        |
| B    BROWSE TRACE DATA   M    MONITOR SCRIPT#  UT    UTILITY MENU           |
|                                                                              |
|Port  System   Terminal Session   Script   Record  Trace                     |
|---- -------- --------  -------- -------- -------- --------                  |
| 1    CICST1   PT01      IN       14 #EX 8                                    |
| 2    CICST1   PT02      OUT       9 #EX 6                                    |
| 3    CICST1   PT03      I/O      11 #EX 7                                    |
| 4                                                                            |
| 5                                                                            |
| 6                                                                            |
| 7                                                                            |
| 8                                                                            |
| 9                                                                            |
| 10                                                                           |
| 11                                                                           |
| 12                                                                           |
|------------------------------------------------------------------------------|
|Copyright (C) 2007, Platinum Technology, Inc. All Rights Reserved. R310       |
+------------------------------------------------------------------------------+
```

When a script ends a session, the status, ENDED, appears in the Session column. When the script ends, ENDED appears in the Script column.

Controlling Multiple Session Tests

When CA Enterprise Tester scripts are executed interactively using the Play Panel, script logs are kept in memory and may be viewed using the Monitor Panel. When a script ends, the log is purged.

Before starting a multiple session test, the CA Enterprise Tester LOG command Log Panel may be used start a log to capture script messages in a data set as described in Script Logs and Diagnostic Traces. When a log is active while scripts are executing, messages displayed in the Monitor Panel Message Log are also written to the log data set. Messages logged while multiple scripts are executing concurrently are logged in chronological order. Extra messages identify the port from which subsequent messages originated.

A log may be written to a named data set or may be written to JES as a SYSOUT data set. When data is written to JES, it is possible to view the log while it is active by using ISPF split-screen to access SDSF or another JES access tool to browse the data set.

After conducting a multiple-session test, a log that is written to a data set may be reordered using the Log Sort Utility, described in The Log Sort Utility, to view all messages for each port separate from messages logged concurrently by other ports.

The TRACE command and Trace Panel may be used to trace individual sessions that are created by the scripts that are used to drive a multiple session test.

While a multiple session test is active, Interactive System commands, entered at the Control Panel, may be used to control the status and speed of active scripts.

- The STOP and RESUME commands may be used to stop or resume the execution of all scripts or a script that is executing on a specified port.
- The EXECUTE command may be used to enter a PSL DELAY PERCENT statement or THINK statement to be executed on a specified port or on all ports, to change the speed of one or all script-driven sessions.
- The THINK RATE statement may be executed to change the total workload delivered by all played scripts. For example, if 50 scripts have been played, the following command may be entered before the RESUME command is entered to begin script execution or while played scripts are active to cause the scripts to generate a total workload of 20 transactions per second:

  ```
  COMMAND ===> ex 'delay percent 0 think rate 20 sessions 50'
  ```

- If a problem occurs while a script is executing, the script is stopped and the status AT ERROR is displayed in the Control Panel Script status column. The MONITOR command may be entered to display the Monitor Panel for a specified port. The Monitor Panel Message Log contains messages that describe the problem. While using the Monitor Panel, the ROLL command may be used to switch directly to the Monitor Panel for the next active port.
- The SESSION command may be used to select a session on a specified port for interactive use. If a script is executing, it is automatically stopped and must be manually resumed.
- When a session is selected, the STEP command may be used to examine the terminal inputs generated by the active script and to view the online system responses. If a non-zero step display time interval has been specified on the Options Panel, the STEP command may specify a number of session steps to automatically display a number of session screens. The RESUME command may be entered to allow an interrupted script to continue execution in the background, at it's own pace.

The behavior of an online system, while a multiple session test is in progress, may best be observed using an independent monitoring tool that is installed in the online system being tested or that monitors the system from another address space. Using another CA Enterprise Tester port, an extra session may be started with the system being tested or with an external monitoring system, to access monitoring system interactive displays.

Other results of an online system test may be obtained from the online system's job log or other log used by an online system to report abnormal completion messages, statistics and other information. Performance data collected by an independent online system monitor and standard online system performance reporting procedures may be used to obtain response times and other performance statistics that may be reliably compared to standard production system performance reports.

Using Select Scripts to Manage Multi-Session Tests

The simple multi-session test described in A Basic Multiple Session Test, was started by entering commands to play different scripts on different Interactive System ports. While the procedure is simple enough when only a few sessions are required for a particular test, other methods are required to configure and start multiple session tests that require a large number of different sessions.

A Play Panel option, Number of Ports, may be used to play a single script on any desired number of ports by entering a single Play Panel request. While a single Play Panel request may specify only a single script, the Number of Ports option may be used to play any combination of different scripts on multiple ports by playing a special script called a select script.

A select script is any script that uses PSL variable, &PTEPORT, to select and link to a different script and to perform other functions, depending on the port on which it is executing. While a script is executing, the value of &PTEPORT is the number of the port on which the script is executing.

For example, the following select script may be played on 10 ports to play 10 scripts named SESS1 through SESS10, at the same time:

```
SET &SCRIPT TO 'SESS' &PTEPORTLINK &SCRIPT
```

The SET statement constructs a script name by concatenating a constant, SESS, and &PTEPORT. When the script is played on 10 ports, a different script name is constructed when the SET statement is executed on each port, and each port links to a different script.

Select scripts may select different scripts for different ports even when script names do not follow an orderly naming convention. For example, the following select script creates a table of 8-character script names to be played on each port, then uses &PTEPORT to locate the script to be played in the table of script names:

```
SET &SCRIPTS TO 'SCRIPTA SCRIPTB SCRIPTC SCRIPTD SCRIPTE'
 'SCRIPTF SCRIPTG SCRIPTH SCRIPTI SCRIPTJ'COMPUTE &SCRIPT = &PTEPORT * 8 - 7LINK
&SCRIPTS(,&SCRIPT,8)
```

In addition to selecting scripts for each port when multi-session tests are started, select scripts may also select and use other kinds of data in the same way that it selects and links to scripts.

For example, select scripts may start sessions, and may select and enter user IDs and passwords to sign on, before linking to application scripts that use readied-sessions to enter application transactions. Structuring scripts in this way provides many possible advantages.

- User ID and password selection is centrally controlled and script maintenance is simplified.
- Application scripts may be recorded by users familiar with application transactions and may be used by anyone authorized to use the transactions to conduct a multi-session online system test.
- Application scripts may be played on more than one port at the same time by signing each port on as a different user ID, possibly reducing the number of scripts required to conduct certain tests.

The following procedure may be used to capture application scripts that are free of user IDs, passwords and session startup logic:

1. Using the SESSION command, start a session with an online system.
2. Sign on using an appropriate user ID and password.
3. Press the ZOOMPF key to re-display the session using Window Mode.
4. Enter the RECORD command, abbreviated r, in the Session Panel command field.
5. At the Record Panel, enter the script library and the name of the script to be recorded and press enter.
6. At the Session Panel, press the ZOOM PF key to re-display the session using the full terminal screen.
7. Enter any application transactions to be recorded.
8. While the session is active, press the ZOOM PF key and enter the END RECORD command, abbreviated e r, in the Session Panel command field.

When a script is played on one or more ports, the speed the scripts may be regulated by entering a delay percent or think time in Play Panel fields. However, using delay percent or fixed think time, it may be difficult to control the total workload delivered by the scripts.

The following logic may be used in a select script to regulate the total workload delivered by scripts played on multiple ports, by entering script parameters at the Play Panel when the select script is played. For example, if a select script that uses the logic is played on 40 ports, RATE=20, PORTS=40 may be entered as script parameters, to generate a total workload of 20 transactions per second.

```
if &rate > 0 and &ports > 0
thendelay percent 0think rate &rate sessions &ports
endif
```

The CA Enterprise Tester Multi-Port Select Utility may be used to create several types of select scripts automatically, including select scripts that generate the names of scripts and scripts that select script names from a table of names created by the utility. The Select Utility provides options that may be used to define other data to be selected for each port, and to set up logic to select the data. Refer to The Multi-Port Select Utility for complete information about and examples select scripts created by the Multi-Port Select Utility.

Creating Scripts with Power and Flex

While CA Enterprise Tester can be used as a simple session capture/play back tool as described in A Basic Multiple Session Test, the CA Enterprise Tester Script Language provides the flexibility to create scripts that can generate any desired number of transactions. In addition to creating large workloads, CA Enterprise Tester scripts are not limited to simply re-keying the data that was captured when a session was recorded. CA Enterprise Tester scripts may obtain data from parameters, data sets, and tables, to create transactions that were not captured and that are not a simple repetition of a previous transaction.

The CA Enterprise Tester Script Language provides statements that allow a script to decide what transaction to enter next based on responses to previously entered transactions. This section illustrates some ideas that may be used to create small scripts that are both powerful and flexible.

The following examples illustrate some techniques that may be used to create powerful and flexible application scripts that may be used to drive multiple session online system tests. It is assumed that a select script is used to start sessions and manage user IDs and passwords, before linking to scripts that enter only business transactions. The recorded and enhanced scripts shown below therefore do not contain LOGON statements or the statements required to enter a user IDs and password to sign on.

Using Loops to Repeat Transactions

A simple technique that may be used to multiply the number of transactions generated by a recorded script is to modify a recorded script to loop. The following script named ISPFTEST, is a recording of the body of a TSO session. When executed on a TSO/ISPF session, the script allocates a new data set, creates a member in the data set, and then deletes the entire data set.

```
TYPE <2,15>   '=3.2'
     <2,18>   ENTER
TYPE <2,15> 'a'
     <15,25>  'test.pds'
     <15,33>  ENTER
TYPE <8,29>   'cyls' ERASE
     <9,29> '1'<10,29>  '1'
     <11,29>  '10'<12,29>  'fb'
     <13,29>  '80'<14,29>  '6160'
     <14,33>  ENTER
TYPE  <2,15>   '=2'
       <2,17>   ENTER
TYPE  <11,25>  'test.pds(member)'
      <11,41>  ENTER
TYPE  <2,15>   '=3.2'
      <4,9>    "this is some test data'"
      <2,19>   ENTER
TYPE  <2,15> 'd'
      <15,25>  'test.pds'
      <15,33>  ENTER
TYPE  <2,15>   ENTER
```

The following modified script allocates the data set, creates the new member, and deletes the entire data set 100 times.

```
set &count to 0
do until &count = 100    set &count up 1
    TYPE <2,15>   '=3.2'
         <2,18>  ENTER
    TYPE <2,15>  'a'
         <15,25>  'test.pds'
         <15,33> ENTER
    TYPE <8,29>   'cyls' ERASE
         <9,29>  '1'
         <10,29> '1'
         <11,29> '10'
         <12,29> 'fb'
         <13,29> '80'
         <14,29>  '6160'
         <14,33> ENTER
    TYPE <2,15>  '=2'
         <2,17>  ENTER
```

```
    TYPE <11,25>  'test.pds(member)'
         <11,41> ENTER
    TYPE <2,15>   '=3.2'
         <4,9>    "this is some test data'"
        <2,19>  ENTER
    TYPE <2,15>  'd'
         <15,25>  'test.pds'
         <15,33> ENTER
    TYPE <2,15>  ENTER
enddo
```

A stress test of several TSO and ISPF functions may be conducted by playing a select script that logs on to TSO, selects and enters a user ID and password, and that displays the ISPF/ PDF Primary Option Menu before linking to the modified script shown above.

As an alternative to modifying an application script to loop, a PSL DO/ENDDO construct may be placed in a select script that links to the recorded ISPF/PDF application script.

For example:

```
set &script to 'ISPFTEST'
set &userid to 'TID' &pteport
set &token  to 'uid1t'
link 'TSOLOGON'
SET &LINK TO 0  /* link to script 100 times */
DO UNTIL &LINK = 100
   LINK &SCRIPT
   SET &LINK UP 1ENDDO
link 'TSOLOGOF'
```

Using Script Parameters

Script parameters may be used to pass information to played scripts. For example, the DO loop described in the previous example could have been coded to LINK to ISPFTEST a variable number of times.

```
.
.
SET &LINK TO 0  /* link to script &LINKS times */
DO UNTIL &LINK = &LINKS
   LINK &SCRIPT
   SET &LINK UP 1ENDDO..
```

When a select script is played, the Play Panel EXEC PARM may be used to assign a value to variable, &LINKS. For example, the following parameter makes the previous select script link to ISPFTEST 25 times.

```
EXEC PARM ===> LINKS=25
```

Multiple parameters may be passed to a script by separating parameter names and values by a comma. For example:

```
EXEC PARM ===> LINKS=25,RATE=30,PORTS=10
```

Parameters that are different for each port may be set up in and selected by a select script. For example, LINKS could be defined as a field name when the Select Utility is used to create a select script, and desired values for each port could be selected by the select script.

Making Scripts Type Variable Data

CA Enterprise Tester variable arrays may be used to store data that may be used for any purpose. For example, the following recorded script, named EMAILINQ selects the directory option on an electronic mail main menu, enters a command to locate a name in the directory, then presses the PF3 key to return to the email menu.

```
TYPE <2,15> 'd'
     <2,16> ENTERTYPE
     <2,15> 'l smith'<2,22> ENTER
TYPE <2,15> PF3
```

Rather than keep typing l smith every time the script is executed in a DO statement loop, the following modified script creates an array of names, the first time it is executed, and uses PTEPORT to select the first name used. Each subsequent use of routine selects the next entry in the array, or selects the first name when the last name was previously used.

```
IF &NAME-I < 1
THEN
   SET &NAME(1)  TO 'smith'
   SET &NAME(2)  TO 'jones'
   SET &NAME(3)  TO 'doe'
   SET &NAME(4)  TO 'winkler'
   SET &NAME(5)  TO 'clinton'
   SET &NAME-I   TO &PTEPORT
ENDIF
COMPUTE &NAME-I = &NAME-I / REMAINDER 5 + 1
TYPE <2,15> 'd'
     <2,16> ENTER
TYPE <2,15> 'l ' &NAME(&NAME-I)
     <2,22> ENTER
TYPE <2,15> PF3
```

Scripts that Read and Use Real Data

CA Enterprise Tester scripts may use Script Language statements to access sequential data sets and VSAM Key Sequenced Data Sets. The PSL READ statement puts a record in a variable field. Record fields, identified by position within a record and field length, may be used by other Script Language statements, including TYPE statements.

For example, assume that an old VSAM name and address file is being converted to a new data base structure, and that a new online data entry transaction has been developed to maintain the data base. Using a CA Enterprise Tester script, the can be read from the old VSAM file and can be loaded into the new data base using the data base maintenance transaction. By using a CA Enterprise Tester script and the new online transaction to load the data base, not only is the data base loaded without writing another program, but also, the new data base maintenance transaction is given a good workout.

The following script illustrates the idea:

```
allocate &names dataset 'vsam.name.and.address' shr
open input &names
logon cics1 24x80 24x80 extds
invite 1000
type 'userid' tab code 'S#FDR' enter
type clear
type 'NAME' enter
type 'add'
do until &eof = 'yes'
   read &names into &record endfile *
   if &pteresp eq '0000'
   then
      set &last-name to &record(,01,20)
      set &first-name to &record(,21,20)
      set &address to &record(,41,20)
      set &city to &record(,61,20)
      set &state-code to &record(,81,02)
      set &zip to &record(,83,05)
      type &last-name
        tab &first-name
        tab &address
        tab &city
        tab &state
        tab &zip enter
      if <1 1 80> contains 'name and address added'
      then
        next
      else
        log 'add failed for ' &record
        dump screen
      endif
        else
        set &eof to 'yes'
      endif
    enddo
      close &names
      free &names
      type clear
      type 'cssf logoff' end
```

Data that is read from a data set may be used in any way. For example, data sets may be accessed by scripts used in multiple session tests. Data that is read into a CA Enterprise Tester variable may be keyed into online system transaction displays or may be compared to data displayed on a virtual terminal screen to verify that expected responses were received.

If a select script is used to play a script that reads and enters data the select script could be set up to select a different initial record number or record key for each port, allowing each copy of the played script to read and process different data.

Unattended Multiple Session Tests

The CA Enterprise Tester Interactive System facilities and techniques described in previous sections allow a person to manually start and control a multiple session test by playing scripts on multiple CA Enterprise Tester ports or by playing a select script on multiple ports to start multiple scripts at the same time. Other commands and panels were described that allow an Interactive System user to monitor and control a multi-session test while it is in progress. While there are many advantages in conducting multi-session tests interactively, it may be desirable to set up a job that may be scheduled for execution at night, to avoid using prime-time system resources.

The CA Enterprise Tester Multi-Port Watch Utility may be used to set up a job that may be submitted to start and watch a multi-session test that is attended by a CA Enterprise Tester script instead of a person. A job created by the Multi-Port Watch Utility uses a distributed CA Enterprise Tester script, PTEMPLAY, which is designed to play a select script and to watch script activity, using a TSO session and the CA Enterprise Tester Interactive System, in exactly the same way that a person would. For complete information about jobs created by the Multi-Port Watch Utility and the PTEMPLAY script, refer to The Multi-Port Watch Utility.

Chapter 9: Utilities

This chapter describes various utilities which are provided to assist in a variety of tasks related to CA Enterprise Tester scripts, traces, REXX execs, and other program products.

- CA Enterprise Tester Utilities
- The Utility Menu

CA Enterprise Tester Utilities

CA Enterprise Tester Utilities are interactive functions that may be selected from the Utility Menu to print or reformat previously captured session data, to set up JCL to run batch applications, to set up scripts to play different scripts on multiple CA Enterprise Tester ports, or to execute and monitor REXX execs that use PSL commands for REXX.

The Utility Menu

The CA Enterprise Tester Utility Menu, shown in the following example, is displayed while using the CA Enterprise Tester Interactive System when the UTILITY command, abbreviated UT, is entered at the Control Panel. The Utility Menu is a menu of interactive utility functions that may be selected by entering the single-character option code displayed preceding the description of each utility function, in the Utility Menu OPTION field.

```
+-------------------------------------------------------------------------+
| ProTermUtility Menu ----------------------------------------------------|
| OPTION ===>                                                             |
|                                                                         |
|       ENTER OPTION                                                      |
| PF3  END TO EXIT MENU                                                   |
| A   Age      Define and age dates in a session trace data set           |
| T   Trace    Print or reformat a session trace data set                 |
| R   Record   Read a trace data set and record in PSL or REXX           |
| L   Log      Sort ProTerm log messages by port number and time          |
| P   Play     Set up a batch job to play a ProTerm script               |
| S   Select   Set up a script to select and link scripts on multiple ports |
| W   Watch    Set up a batch job to play and watch scripts on multiple ports |
| C   Compare  Set up a batch job to compare session trace data sets     |
| M Monitor    Execute and monitor REXX execs and exec-driven sessions   |
+-------------------------------------------------------------------------+
```

Utility Options

- The Age option selects the Session Ager. The Session Ager is an interactive utility that ages dates in session data that was previously captured using the Trace facility. Using aged test sessions created using the Session Ager, aged test data created using CA FileAge product and session analyzers, online software tests performed manually using a particular current system date to verify date-sensitive compliance, may be repeated and analyzed automatically using other current system dates. Refer to The Session Ager for complete documentation.
- The Trace option selects the Trace Utility. The Trace Utility reads a session trace data set that was previously captured using the trace Format option, and creates a new trace data set using specified trace options. The Trace utility may be used to print virtual terminal screen images captured for regression testing, may be used to add DELAY statements to a session trace data set that was originally captured without delays, or may be used to reformat traced session data using other options similar to the trace options available when an interactive session is originally traced. Refer to The Trace Utility for complete documentation.
- The Record option selects the Record Utility. The Record Utility reads a session trace data set that was previously captured using the trace Format option, and creates a script or REXX exec that creates a session similar to the traced session when the script or exec is executed. When a trace is recorded as a script or REXX exec, user keystrokes and other session events captured as traced screen images are recorded as PSL statements or commands for REXX. A script or exec created by the Record Utility is similar to a script or exec that would have been created if the traced session were originally recorded as a script or exec. Refer to The Record Utility for complete documentation.
- The Log option selects the Log Sort Utility. The Log Sort Utility reads a log data set containing chronological messages issued by multiple scripts that were executed concurrently on multiple ports, for example, to drive a multi-session online system test, and reorders the messages into groups containing all messages for each port. The Log Sort Utility may be used to print or write a reordered log to a data set to view all messages related to events that occurred on each port separate from other messages that document events that occurred concurrently on other ports. Refer to The Log Sort Utility for complete documentation.
- The Play option selects the Batch Play Utility. The Batch Play Utility sets up a batch job using specified data sets, options and script parameters, that may be submitted to play a CA Enterprise Tester script in batch. Refer to The Batch Play Utility for complete documentation.

- The Select option selects the Multi-Port Select Utility. The Multi-Port Select Utility sets up a script that selects and links to a script and selects other user-defined data, depending on which port the script is executing. A script created by the Multi-Port Select Utility may be played on multiple ports using the Play Panel number of ports option to play different scripts on different ports and may be used to start flexible multi-session online system tests by entering a single play command. Refer to The Multi-Port Select Utility for complete documentation.
- The Watch option selects the Multi-port Watch Utility. The Multi-Port Watch Utility sets up a job that starts a TSO session and uses the Interactive System Play Panel to play a script on multiple ports, for example a script created using the Multi-Port Select Utility. After playing the script on the specified number of ports, the job uses the Control Panel and other Interactive System functions to watch script activity and to document errors and to otherwise manage a multi-port application. A job created by the Multi-Port Watch Utility may be scheduled to run during off-hours to drive a multi-session online system test when system resources are more available. Refer to The Multi-Port Watch Utility for complete documentation.
- The Compare option selects the Batch Compare Utility. The Batch Compare Utility sets up a batch job that may be submitted to compare two session trace data sets. The script that compares the traced sessions uses mask scripts and reports mismatches using techniques used by session analyzers when a live session is created and compared to previously traced session. Refer to The Batch Compare Utility for complete documentation.
- The Monitor option selects the Command Monitor for REXX. The Command Monitor for REXX executes one or more REXX execs that use CA Enterprise Tester commands and displays exec-driven events and sessions at a terminal while the exec or execs are running. Multiple execs may be executed to drive a multi-session online system test and to monitor exec-driven events and sessions that are displayed in windows, allowing multiple concurrent exec-driven events to be viewed at a single terminal at the same time while a test is in progress. Refer to The CA Enterprise Tester Command Monitor for REXX for complete documentation.

The Session Ager

When testing date-dependent software functions in date-sensitive applications, the date logic must be tested multiple times using current system dates to verify that both past and future dates including the current date are processed correctly by the application. Each time testing is repeated using a different current system date, dates in test data may need to be correspondingly aged to assure that tests of computations and decisions that use the dates and the current date are equivalent each time the function tests are repeated.

The Session Ager is an interactive utility that creates aged versions of traced sessions that were captured using the trace Format option. CA Enterprise Tester session analyzers and aged sessions may be used to repeat date-sensitive compliance testing automatically when testing is repeated using different current system dates. An aged session is created by adding a specified number of days to dates in captured screen images that contain the data entered as input to each test transaction or the information displayed in each transaction response. When an aged session is repeated using a CA Enterprise Tester session analyzer, the recalculated input dates are entered instead of the captured input dates and the recalculated response dates are compared to dates displayed in response to each repeated transaction.

The Session Ager uses screen definitions to determine the layout of each captured screen image processed and to locate and interpret the dates in each defined screen that are to be aged. However, screen definitions are not required to begin aging captured sessions. Instead, they are created when needed during Session Ager processing. When the Session Ager encounters an undefined screen image, the screen image and a series of step-by-step instructions are displayed that explain to the user how to point to key information that indicates the layout of the screen image and how to identify and define dates that are to be aged using pictures that define the format of dates. The Session Ager keeps track of the information provided in a screen definition data set and uses it to age a defined screen image and all similar screen images that are subsequently processed. The screen definition data set may be used and extended when other sessions are subsequently aged, and may be shared by a group of users to reuse screen definitions created by anyone in the group.

Aging Captured Sessions

To use the Session Ager utility, enter the UTILITY command, abbreviated UT, at the Control Panel to display the Utility Menu, then select Utility Menu option A to display the Session Ager request panel in the following example.

```
+-------------------------------------------------------------------------+
| ProTermSession Ager ---------------------------------------------------|
| COMMAND ===>                                                            |
|                                                                         |
|   ENTER TO AGE DATA                                                     |
| PF3  END TO EXIT PANEL                                                  |
|                                                                         |
| Specify session trace or trace list to be aged and ager options:        |
|                                                                         |
|   Trace Library ===> PROTERM.TRACE                                      |
|   Member Name   ===> agetest1 ( Blank or pattern for member selection list)
|                               ( Select codes: S Select, B Browse, E Edit ) |
|   Days to Add   ===> 365      ( Number of days to age defined trace dates ) |
|   Lowest 19YY   ===> 00       ( ie., if 70-69 means 1970-2069, specify 70 ) |
|                                                                         |
|                                                                         |
| Specify target libraries for aged session trace(s) and ager message log:|
|                                                                         |
|   Trace Library ===> PROTERM.AGETRACE                                   |
|   Log Library   ===> PROTERM.LOG                                        |
|                                                                         |
| Specify sequential date definition trace data set:                      |
|                                                                         |
|   Date Def DSN  ===> 'GROUP.PROTERM.AGERDEFS'                           |
+-------------------------------------------------------------------------+
```

To request Session Ager services, enter the data sets and option values to be used by the Session Ager, as follows:

To display a Trace Library member selection list:

- Enter the name of a PDS or library as the Trace Library to be aged.
- Enter blanks or a member name pattern as the Member Name to be aged.
- Press ENTER.

While using a member selection list, members may be selected, browsed or edited using S, B or E member list select codes.

To create aged versions of one or more previously captured sessions:

1. Enter the Trace Library and Member Name to be aged. The library member specified may be a CA Enterprise tester session trace that was captured using the trace Format option, or may be a session analyzer Trace List that contains one or more

ANALYZE statements that specify the names of session traces to be aged. For example, a PDS member containing the following Trace List statements could be specified as the Member Name to be aged to age the three traced sessions specified.

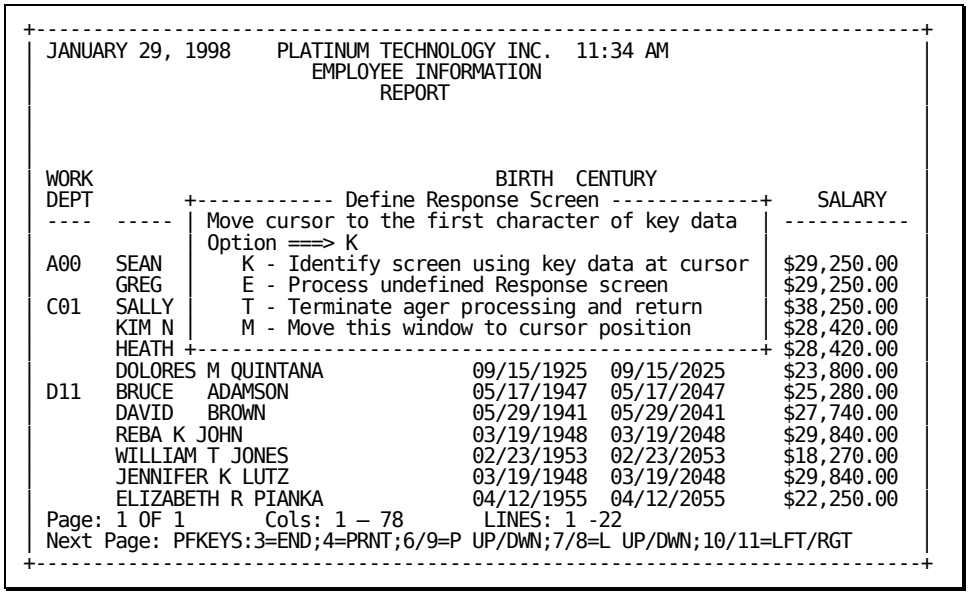
```
ANALYZE=UID.AGETRACE(SESSION1)
ANALYZE=UID.AGETRACE(SESSION2)
ANALYZE=UID.AGETRACE(SESSION3)
```

Refer to The CA Enterprise Tester Session Analyzer for a more detailed description of the session analyzer trace lists.

2. Enter the number of Days to Add to defined dates found in the session being aged to calculate the aged dates stored in the aged version of the session. The number of days may be either a positive integer or negative integer with a leading minus sign.
3. Enter the Lowest 19YY to be used by the Session Ager to determine the century of dates with 2-digit years. A 2-digit year that is greater than or equal to the 2-digit year specified is interpreted as a year in the 1900s. A 2-digit year that is less than the 2-digit year specified is interpreted as a year in the 2000s.
4. Enter the name of the target Trace Library for aged versions of session(s) to be created. The member name of each session to be aged is used as the member name of the corresponding aged session written to the target Trace Library. To avoid replacing sessions to be aged, create and specify a new target trace library with RECFM=FB and LRECL=135.
5. Enter the name of the target Log Library used by the Session Ager to log messages that document any processing errors that may occur. The Member Name in the Trace Library to be aged is also used as the member name of the message log written to the target Log Library.
6. Enter the name of the Date Def DSN to be used to save and retrieve screen definitions created during Session Ager processing. A Session Ager screen definition data set is a sequential data set with RECFM=FB and LRECL=135. Either specify the name of an existing screen definition data set to reuse screen definitions saved during previous Session Ager processing, or create and specify a new empty screen definition data set if a suitable screen definition data is not available. Any new screen definitions created during Session Ager processing will be added to the definition data set specified. A single screen definition data set may be shared by multiple users to reuse screen definitions created by other users.
7. Press ENTER.

Defining Screen Image Keys

When the Session Ager processes an undefined screen image, the undefined screen image and a window containing instructions to point to key data displayed in the screen image using the cursor. Every screen image definition must identify one or more key values that the Session Ager may use to select the definition that applies to all similar screen images. The cursor is moved to the first character of the screen image title, EMPLOYEE INFORMATION, before pressing ENTER, to define the value as a screen image key.



Following instructions in the Define Key Data Length window the cursor is moved to the last character of EMPLOYEE INFORMATION, before pressing ENTER, to identify the end of the key data.

```
+-------------------------------------------------------------------------------+
| JANUARY 29, 1998    PLATINUM TECHNOLOGY INC.  11:34 AM                        |
|                        EMPLOYEE INFORMATION                                   |
|                              REPORT                                           |
|                                                                               |
|                                                                               |
| WORK                                        BIRTH  CENTURY                    |
| DEPT        +------------ Define Response Screen -------------+    SALARY    |
| ----  ----- | Move cursor to the first character of key data  | ----------    |
|             | Option ===> K                                   |               |
| A00   SEAN  |    K - Identify screen using key data at cursor | $29,250.00    |
|       GREG  |    E - Process undefined Response screen        | $29,250.00    |
| C01   SALLY |    T - Terminate ager processing and return     | $38,250.00    |
|       KIM N |    M - Move this window to cursor position      | $28,420.00    |
|       HEATH +-------------------------------------------------+ $28,420.00    |
|       DOLORES M QUINTANA             09/15/1925  09/15/2025     $23,800.00    |
| D11   BRUCE   ADAMSON                05/17/1947  05/17/2047     $25,280.00    |
|       DAVID   BROWN                  05/29/1941  05/29/2041     $27,740.00    |
|       REBA K JOHN                    03/19/1948  03/19/2048     $29,840.00    |
|       WILLIAM T JONES                02/23/1953  02/23/2053     $18,270.00    |
|       JENNIFER K LUTZ                03/19/1948  03/19/2048     $29,840.00    |
|       ELIZABETH R PIANKA             04/12/1955  04/12/2055     $22,250.00    |
| Page: 1 OF 1       Cols: 1 - 78       LINES: 1 -22                           |
| Next Page: PFKEYS:3=END;4=PRNT;6/9=P UP/DWN;7/8=L UP/DWN;10/11=LFT/RGT        |
+-------------------------------------------------------------------------------+
```

When a key data value has been defined, the Define Key Data Location options window provides options that allow screen image keys to be defined that may be displayed at other than a fixed screen location.

```
+-------------------------------------------------------------------------------+
| JANUARY 29, 1998    PLATINUM TECHNOLOGY INC.  11:34 AM                        |
|                        EMPLOYEE INFORMATION                                   |
|                              REPORT                                           |
|                                                                               |
|                                                                               |
| WORK                                        BIRTH  CENTURY                    |
| DEPT        +------------ Define Response Screen -------------+    SALARY    |
| ----  ----- | Move cursor to the first character of key data  | ----------    |
|             | Option ===> K                                   |               |
| A00   SEAN  |    K - Identify screen using key data at cursor | $29,250.00    |
|       GREG  |    E - Process undefined Response screen        | $29,250.00    |
| C01   SALLY |    T - Terminate ager processing and return     | $38,250.00    |
|       KIM N |    M - Move this window to cursor position      | $28,420.00    |
|       HEATH +-------------------------------------------------+ $28,420.00    |
|       DOLORES M QUINTANA             09/15/1925  09/15/2025     $23,800.00    |
| D11   BRUCE   ADAMSON                05/17/1947  05/17/2047     $25,280.00    |
|       DAVID   BROWN                  05/29/1941  05/29/2041     $27,740.00    |
|       REBA K JOHN                    03/19/1948  03/19/2048     $29,840.00    |
|       WILLIAM T JONES                02/23/1953  02/23/2053     $18,270.00    |
|       JENNIFER K LUTZ                03/19/1948  03/19/2048     $29,840.00    |
|       ELIZABETH R PIANKA             04/12/1955  04/12/2055     $22,250.00    |
| Page: 1 OF 1       Cols: 1 - 78       LINES: 1 -22                           |
| Next Page: PFKEYS:3=END;4=PRNT;6/9=P UP/DWN;7/8=L UP/DWN;10/11=LFT/RGT        |
+-------------------------------------------------------------------------------+
```

The options window displayed after the first screen image key is defined provides the option to define additional keys, so that screen image layouts that can not be uniquely identified using a single key value may be defined.

Defining Date Fields and Formats

The Session Ager ages dates only if they are defined. If a screen image contains dates that are to be aged, each date to be aged must be defined using the date definition option displayed in the screen definition options window after at least one screen image key has been defined. A date is defined by entering the date definition option, D, in the screen definition window options field before moving the cursor to the first position of the date field to be defined. In the following example, the cursor is positioned at the J in January 29, 1998, displayed in row 1 of the screen image being defined.

```
+-------------------------------------------------------------------------------+
| JANUARY 29, 1998    PLATINUM TECHNOLOGY INC.  11:34 AM                        |
|                        EMPLOYEE INFORMATION                                   |
|                              REPORT                                           |
|                                                                               |
|                                                                               |
| WORK                                        BIRTH  CENTURY                    |
| DEPT        +------------ Define Response Screen -------------+    SALARY    |
| ----  ----- | Move cursor to the first character of key data  | ----------     |
|             | Option ===> K                                   |                |
| A00   SEAN  |    K - Identify screen using key data at cursor | $29,250.00     |
|       GREG  |    E - Process undefined Response screen        | $29,250.00     |
| C01   SALLY |    T - Terminate ager processing and return     | $38,250.00     |
|       KIM N |    M - Move this window to cursor position      | $28,420.00     |
|       HEATH +-------------------------------------------------+ $28,420.00     |
|       DOLORES M QUINTANA             09/15/1925  09/15/2025     $23,800.00     |
| D11   BRUCE   ADAMSON                05/17/1947  05/17/2047     $25,280.00     |
|       DAVID   BROWN                  05/29/1941  05/29/2041     $27,740.00     |
|       REBA K JOHN                    03/19/1948  03/19/2048     $29,840.00     |
|       WILLIAM T JONES                02/23/1953  02/23/2053     $18,270.00     |
|       JENNIFER K LUTZ                03/19/1948  03/19/2048     $29,840.00     |
|       ELIZABETH R PIANKA             04/12/1955  04/12/2055     $22,250.00     |
| Page: 1 OF 1       Cols: 1 – 78       LINES: 1 -22                            |
| Next Page: PFKEYS:3=END;4=PRNT;6/9=P UP/DWN;7/8=L UP/DWN;10/11=LFT/RGT        |
+-------------------------------------------------------------------------------+
```

The format of a date is defined by entering a picture of the date in the modifiable date field displayed when the date definition option is used to identify a date field to be defined. A date picture is a string of mnemonics that define the format of each component of the date, and includes blanks and symbols that separate the components of the date. Date picture mnemonics are displayed in the date definition instruction window illustrated in the example shown in the following example.

In the example, the date picture, mnd, yyyy is entered over January 29, 1998, to specify that the date begins with the spelled month name, one blank, the day of month without a leading blank or zero, followed by a comma and one blank, and the 4-digit year.

```
+-------------------------------------------------------------------------------+
|  mn d, yyyy, 1998       PLATINUM TECHNOLOGY INC.        11:34 AM              |
|                           EMPLOYEE INFORMATION                                |
|                                   REPORT                                      |
|                                                                               |
|                                                                               |
| WORK                                              BIRTH  CENTURY              |
|  DEPT        +---------- Define Response Date Picture -----------+ SALARY     |
| -----  ----  | Enter date picture in field(s) at <1 3>           | --------    |
|              | Option ===> D                                     |             |
| A00    SEAN  |   D - Define date picture entered at <1 3>        |$29,250.00   |
|       GREG   |   E - Cancel this date definition                 |$29,250.00   |
| C01    SALLY |   M - Move this window to cursor position         |$38,250.00   |
|       KIM N  | Example: picture mm/dd/yyyy defines 12/31/1999    |$28,420.00   |
|       HEATH  |    m    1-12      d    1-31 yy     00-99         |$28,420.00   |
|       DOLOR  |    mm   01-12     dd   01-31    yyyy 0001-9999   |$23,800.00   |
| D11    BRUCE |    zm    1-12     zd    1-31                     |$25,280.00   |
|       DAVID  |    mon  Jan-Dec   ddd  001-366     wn Monday      |$27,740.00   |
|       REBA   |    mn   January   day  1=1/1/0001 wee Mon        |$29,840.00   |
|       WILLI +-------------------------------------------------+$18,270.00   |
|       JENNIFER K LUTZ                  03/19/1948  03/19/2048   $29,840.00   |
|       ELIZABETH R PIANKA               04/12/1955  04/12/2055   $22,250.00   |
| Page: 1 OF 1          Cols: 1 - 78                 LINES: 1 -22               |
| Next Page: PFKEYS:3=END;4=PRNT;6/9=P UP/DWN;7/8=L UP/DWN;10/11=LFT/RGT        |
+-------------------------------------------------------------------------------+
```

Variable length dates—for example, a date that includes the spelled-out name of the month—are sometimes displayed in variable length fields that are delimited by attributes or pad characters, or that contain other data that is concatenated to a particular variable length date. When a variable length date is aged and the length of the aged date is shorter or longer than the date that was aged, the Session Ager must pad shorter aged dates or shift attributes or concatenated data left or right in the aged screen image, and therefore needs to know the length of the field to be reformatted including any concatenated attributes, pad characters or data to be shifted.

When a variable length date picture is entered, the length of a field is obtained from the user by positioning the cursor at last position of the minimum-length field that could contain the maximum-length date for the date picture specified and displaying instructions shown in the following illustration, that allow the user to define a larger date field.

In most cases, including the example in the following example, the cursor position calculated by the Session Ager correctly identifies the end of the date field and the user may simply press ENTER to define the calculated date field length. A longer date field is defined by moving the cursor to the last position of the date field to be reformatted when the defined variable length dates are aged.

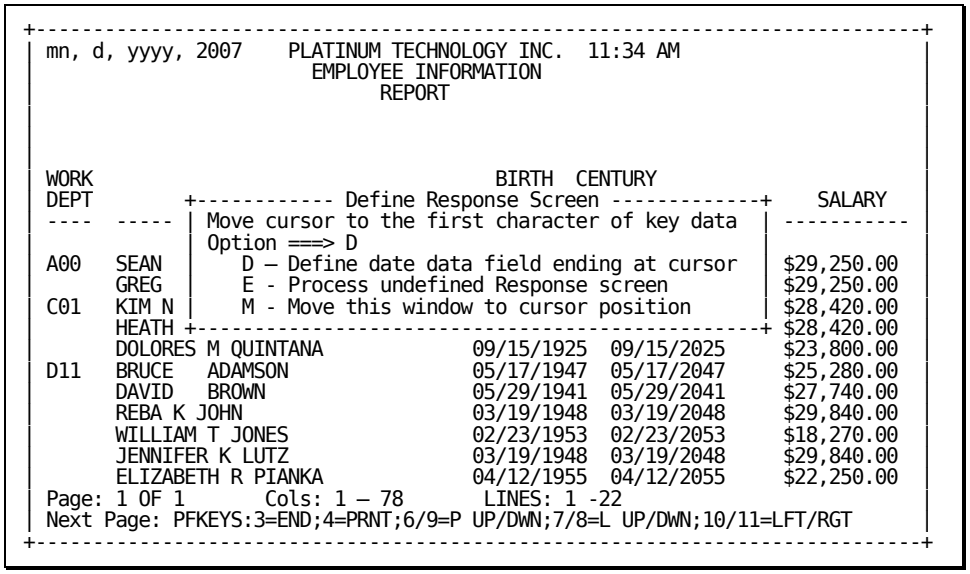
```
+------------------------------------------------------------------------------+
| mn, d, yyyy, 2007    PLATINUM TECHNOLOGY INC.  11:34 AM                      |
|                        EMPLOYEE INFORMATION                                  |
|                              REPORT                                          |
|                                                                              |
|                                                                              |
| WORK                                        BIRTH  CENTURY                   |
| DEPT        +------------ Define Response Screen -------------+    SALARY   |
| ----  ----- | Move cursor to the first character of key data  | ---------- |
|             | Option ===> D                                   |            |
| A00   SEAN  |    D - Define date data field ending at cursor  | $29,250.00 |
|       GREG  |    E - Process undefined Response screen        | $29,250.00 |
| C01   KIM N |    M - Move this window to cursor position      | $28,420.00 |
|       HEATH +-------------------------------------------------+ $28,420.00 |
|       DOLORES M QUINTANA             09/15/1925  09/15/2025     $23,800.00 |
| D11   BRUCE   ADAMSON                05/17/1947  05/17/2047     $25,280.00 |
|       DAVID   BROWN                  05/29/1941  05/29/2041     $27,740.00 |
|       REBA K JOHN                    03/19/1948  03/19/2048     $29,840.00 |
|       WILLIAM T JONES                02/23/1953  02/23/2053     $18,270.00 |
|       JENNIFER K LUTZ                03/19/1948  03/19/2048     $29,840.00 |
|       ELIZABETH R PIANKA             04/12/1955  04/12/2055     $22,250.00 |
| Page: 1 OF 1       Cols: 1 - 78       LINES: 1 -22                         |
| Next Page: PFKEYS:3=END;4=PRNT;6/9=P UP/DWN;7/8=L UP/DWN;10/11=LFT/RGT       |
+------------------------------------------------------------------------------+
```

The Define Date or Date List options window displayed after a date has been defined may be used to ditto the new date definition to subsequent screen rows when it is saved. To save the definition of a date that is not a part of a list of dates, for example, the date defined in the following example, press ENTER.

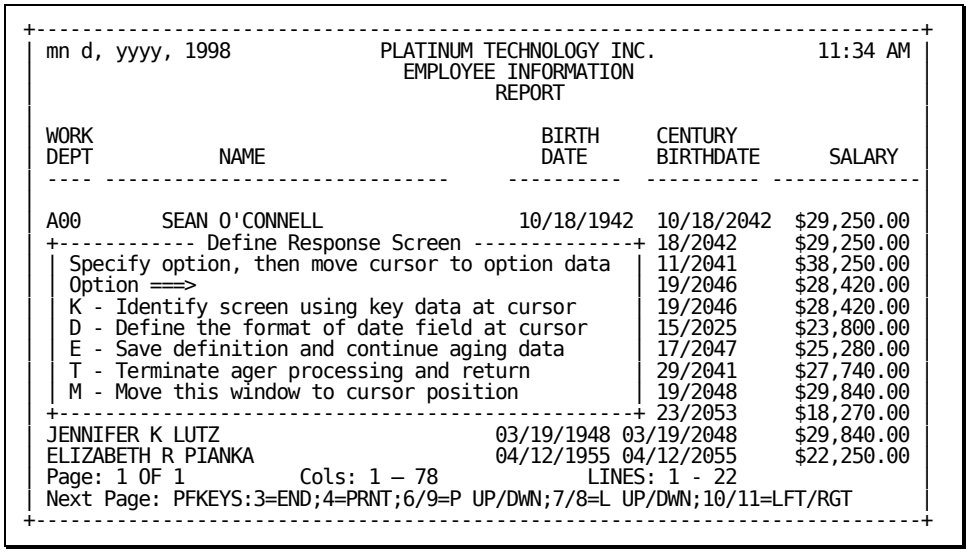
```
+------------------------------------------------------------------------------+
| mn, d, yyyy, 2007    PLATINUM TECHNOLOGY INC.  11:34 AM                      |
|                        EMPLOYEE INFORMATION                                  |
|                              REPORT                                          |
|                                                                              |
| WORK                                        BIRTH  CENTURY                   |
| DEPT        +------------ Define Response Screen -------------+    SALARY   |
| ----  ----- | Move cursor to the first character of key data  | ---------- |
|             | Option ===> D                                   |            |
| A00   SEAN  |    D - Define date data field ending at cursor  | $29,250.00 |
|       GREG  |    E - Process undefined Response screen        | $29,250.00 |
| C01   KIM N |    M - Move this window to cursor position      | $28,420.00 |
|       HEATH +-------------------------------------------------+ $28,420.00 |
|       DOLORES M QUINTANA             09/15/1925  09/15/2025     $23,800.00 |
| D11   BRUCE   ADAMSON                05/17/1947  05/17/2047     $25,280.00 |
|       DAVID   BROWN                  05/29/1941  05/29/2041     $27,740.00 |
|       REBA K JOHN                    03/19/1948  03/19/2048     $29,840.00 |
|       WILLIAM T JONES                02/23/1953  02/23/2053     $18,270.00 |
|       JENNIFER K LUTZ                03/19/1948  03/19/2048     $29,840.00 |
|       ELIZABETH R PIANKA             04/12/1955  04/12/2055     $22,250.00 |
| Page: 1 OF 1       Cols: 1 - 78       LINES: 1 -22                         |
| Next Page: PFKEYS:3=END;4=PRNT;6/9=P UP/DWN;7/8=L UP/DWN;10/11=LFT/RGT       |
+------------------------------------------------------------------------------+
```

Moving Instruction Windows

Session Ager instruction windows cover portions of a screen image and may hide screen image keys or dates that must be defined. All instruction windows provide the option to move the window elsewhere to uncover hidden portions of the screen image. In the screen illustrated in --Heading id 'utamw1' unknown --, the top of a list of dates is hidden by an instruction window. The window may be moved by entering M in the instruction window Option field and by moving the cursor to a new Option field screen location.



The screen image in -- Heading id 'utamw2' unknown-- shows an instruction window that has been moved to a location that allows the first date in a list of dates to be defined that was previously hidden by the instruction window.



Defining Date Lists

When a new date definition is saved an option is provided that may be used to ditto the new definition on subsequent screen rows automatically. The ditto function is designed to define an entire list of similar dates displayed on successive screen rows in one step instead of defining each date separately. To use the ditto function, identify and define the first date in the list of dates by entering the date picture over the date. In the screen shown in the following example, the first date in the list titled BIRTH DATE is defined by entering the date picture, mm/dd/yyyy, over the date, 10/18/1942.

```
+-------------------------------------------------------------------------------+
| mn d, yyyy,            1998 PLATINUM TECHNOLOGY INC.              11:34 AM |
|                                EMPLOYEE INFORMATION                           |
|                                       REPORT                                  |
|                                                                               |
|                                                                               |
| WORK                                        BIRTH    CENTURY                  |
| DEPT          NAME                          DATE     BIRTHDATE     SALARY     |
| ---- ------------------------------ ---------- ---------- --------------     |
|                                                                               |
| A00      SEAN O'CONNELL             mm/dd/yyyy         10/18/2042 $29,250.00 |
| +---------- Define Response Date Picture -----------+ 8/2042     $29,250.00 |
| | Enter date picture in field(s) at <11 40>          | 1/2041     $38,250.00 |
| | Option ===> D                                      | 9/2046     $28,420.00 |
| | D - Define date picture entered at <11 40>         | 9/2046     $28,420.00 |
| | E - Cancel this date definition                    | 5/2025     $23,800.00 |
| | M - Move this window to cursor position            | 7/2047     $25,280.00 |
| | Example: picture mm/dd/yyyy defines 12/31/1999     | 9/2041     $27,740.00 |
| | m     1-12    d    1-31    yy 00-99               | 9/2048     $29,840.00 |
| | mm    01-12   dd   01-31   yyyy 0001-9999         | 3/2053     $18,270.00 |
| | zm    _1-12   zd   _1-31                           | 9/2048     $29,840.00 |
| | mon   Jan-Dec ddd  001-366 wn Monday               | 2/2055     $22,250.00 |
| | mn    January day  1=1/1/0001 wee Mon              | 2                     |
| +----------------------------------------------------+ 8=LP/DWN;10/11=LFT/RGT|
+-------------------------------------------------------------------------------+
```

When the first date in a list of dates has been defined, the Define Date or Date List options window provides an option to ditto the definition to subsequent dates in the list. In the following example , the remainder of the dates in the list are defined and saved all at one time, by moving the cursor to the last list row, which contains the date 04/12/1955, before pressing ENTER.

```
+------------------------------------------------------------------------------+
| mn d, yyyy,          1998 PLATINUM TECHNOLOGY INC.              11:34 AM     |
|                        EMPLOYEE INFORMATION                                  |
|                                REPORT                                        |
|                                                                              |
| WORK                              BIRTH                 CENTURY              |
| DEPT    NAME                       DATE                BIRTHDATE    SALARY   |
| ---- ------------------------------ ---------- ---------- --------------     |
|                                                                              |
| A00     SEAN O'CONNELL            mm/dd/yyyy          10/18/2042  29,250.00  |
| +-------------- Define Date or Date List ----------------+ 2   $29,250.00  |
| | To ditto picture, move cursor down to last date row    | 1   $38,250.00  |
| | Option ===> D                                          | 6   $28,420.00  |
| | D - Save or ditto picture at <11 40> to cursor row     | 6   $28,420.00  |
| | E - Cancel this date definition                        | 5   $23,800.00  |
| | M - Move this window to cursor position                | 7   $25,280.00  |
| +--------------------------------------------------------+ 1   $27,740.00  |
|          REBA K JOHN                    03/19/1948 03/19/2048   $29,840.00  |
|          WILLIAM T JONES                02/23/1953 02/23/2053   $18,270.00  |
|          JENNIFER K LUTZ                03/19/1948 03/19/2048   $29,840.00  |
|          ELIZABETH R PIANKA             04/12/1955 04/12/2055   $22,250.00  |
| Page: 1 OF 1 Cols: 1 - 78                          LINES: 1 - 22             |
| Next Page: PFKEYS:3=END;4=PRNT;6/9=P UP/DWN;7/8=L UP/DWN;10/11=LFT/RGT       |
+------------------------------------------------------------------------------+
```

The date pictures displayed in place of the defined list of dates, like those shown in the following example, confirm that the new date list definition has been saved.

```
+------------------------------------------------------------------------------+
| mn d, yyyy,          1998 PLATINUM TECHNOLOGY INC.              11:34 AM     |
|                        EMPLOYEE INFORMATION                                  |
|                                REPORT                                        |
|                                                                              |
| WORK                              BIRTH                 CENTURY              |
| DEPT    NAME                       DATE                BIRTHDATE    SALARY   |
| ---- ------------------------------ ---------- ---------- --------------     |
|                                                                              |
| A00     SEAN O'CONNELL            mm/dd/yyyy          10/18/2042  29,250.00  |
| +-------------- Define Date or Date List ----------------+ 2   $29,250.00  |
| | To ditto picture, move cursor down to last date row    | 1   $38,250.00  |
| | Option ===> D                                          | 6   $28,420.00  |
| | D - Save or ditto picture at <11 40> to cursor row     | 6   $28,420.00  |
| | E - Cancel this date definition                        | 5   $23,800.00  |
| | M - Move this window to cursor position                | 7   $25,280.00  |
| +--------------------------------------------------------+ 1   $27,740.00  |
|          REBA K JOHN                    03/19/1948 03/19/2048   $29,840.00  |
|          WILLIAM T JONES                02/23/1953 02/23/2053   $18,270.00  |
|          JENNIFER K LUTZ                03/19/1948 03/19/2048   $29,840.00  |
|          ELIZABETH R PIANKA             04/12/1955 04/12/2055   $22,250.00  |
| Page: 1 OF 1 Cols: 1 - 78                          LINES: 1 - 22             |
| Next Page: PFKEYS:3=END;4=PRNT;6/9=P UP/DWN;7/8=L UP/DWN;10/11=LFT/RGT       |
+------------------------------------------------------------------------------+
```

The screen definition options window is displayed after each screen image key or date definition has been completed or canceled, and showing all keys and dates that have been defined. In the event an incorrect key or date definition has been entered, the incorrect key or date definition may be corrected by entering the key or date definition option and by using the cursor to identify the incorrect definition. A key or date definition identified in this way is canceled and a new definition may be entered.

When all key and date definitions have been entered, the end option, E, as shown in the following example, or a PF key assigned to the END command may be used to save the definition in the screen image definition data set and to resume Session Ager processing.

```
+-------------------------------------------------------------------------+
| mn d, yyyy,         1998 PLATINUM TECHNOLOGY INC.             11:34 AM  |
|                       EMPLOYEE INFORMATION                              |
|                                REPORT                                   |
|                                                                         |
| WORK                              BIRTH               CENTURY           |
| DEPT    NAME                       DATE              BIRTHDATE    SALARY |
| ---- ------------------------------ ---------- ---------- -------------- |
|                                                                         |
| A00     SEAN O'CONNELL           mm/dd/yyyy         10/18/2042  29,250.00 |
| +------------- Define Date or Date List ----------------+ 2   $29,250.00 |
| | To ditto picture, move cursor down to last date row    | 1   $38,250.00 |
| | Option ===> D                                          | 6   $28,420.00 |
| | D - Save or ditto picture at <11 40> to cursor row     | 6   $28,420.00 |
| | E - Cancel this date definition                        | 5   $23,800.00 |
| | M - Move this window to cursor position                | 7   $25,280.00 |
| +--------------------------------------------------------+ 1   $27,740.00 |
|         REBA K JOHN                   03/19/1948 03/19/2048   $29,840.00 |
|         WILLIAM T JONES               02/23/1953 02/23/2053   $18,270.00 |
|         JENNIFER K LUTZ               03/19/1948 03/19/2048   $29,840.00 |
|         ELIZABETH R PIANKA            04/12/1955 04/12/2055   $22,250.00 |
| Page: 1 OF 1 Cols: 1 - 78                        LINES: 1 - 22          |
| Next Page: PFKEYS:3=END;4=PRNT;6/9=P UP/DWN;7/8=L UP/DWN;10/11=LFT/RGT  |
+-------------------------------------------------------------------------+
```

Defining Input Screens and Dates

A CA Enterprise Tester session trace captures an input screen image when a terminal user enters data and captures a response screen image when data is displayed by online software. Screens are formatted only by online systems and therefore response screens and the subsequent input screens actually have the same screen format. However, because an input screen may contain input dates that were not apparent in the previous response screen, the Session Ager creates a screen image definition for the first undefined response screen image processed, and creates a second screen definition for the subsequent undefined input screen image to provide an opportunity to define input dates. When an input screen definition is created, all keys and dates defined when the previous response screen definition was created are copied to the new input screen definition so that they do not have to be defined again. While an input screen image definition is being defined, additional dates may be defined, and key or date definitions may be redefined to correct any incorrect definitions in the copied response screen definition.

An input screen definition that is identical to the previous response screen definition may be completed and saved by simply entering option E, as shown in the following example, or by pressing a PF key that is assigned to the END command.

```
+-------------------------------------------------------------------------------+
| mn d, yyyy,         1998 PLATINUM TECHNOLOGY INC.             11:34 AM    |
|                       EMPLOYEE INFORMATION                                    |
|                                REPORT                                         |
|                                                                               |
| WORK                              BIRTH                 CENTURY               |
| DEPT    NAME                       DATE                BIRTHDATE    SALARY   |
| ---- ------------------------------ ---------- ---------- --------------     |
|                                                                               |
| A00     SEAN O'CONNELL           mm/dd/yyyy         10/18/2042  29,250.00 |
| +-------------- Define Date or Date List ----------------+ 2   $29,250.00 |
| | To ditto picture, move cursor down to last date row    | 1   $38,250.00 |
| | Option ===> D                                          | 6   $28,420.00 |
| | D - Save or ditto picture at <11 40> to cursor row     | 6   $28,420.00 |
| | E - Cancel this date definition                        | 5   $23,800.00 |
| | M - Move this window to cursor position                | 7   $25,280.00 |
| +--------------------------------------------------------+ 1   $27,740.00 |
|         REBA K JOHN                   03/19/1948 03/19/2048   $29,840.00 |
|         WILLIAM T JONES               02/23/1953 02/23/2053   $18,270.00 |
|         JENNIFER K LUTZ               03/19/1948 03/19/2048   $29,840.00 |
|         ELIZABETH R PIANKA            04/12/1955 04/12/2055   $22,250.00 |
| Page: 1 OF 1 Cols: 1 - 78                        LINES: 1 - 22              |
| Next Page: PFKEYS:3=END;4=PRNT;6/9=P UP/DWN;7/8=L UP/DWN;10/11=LFT/RGT      |
+-------------------------------------------------------------------------------+
```

Session Ager Screen Definitions

A Session Ager screen definition data set is a sequential data set with RECFM=FB and LRECL=135, which is empty or contains one or more screen definitions supported by the Session Ager. A Session Ager screen definition consists of a screen image created using the trace Format option, and required and optional Session Ager screen definition statements which follow the defined screen image.

One or more Session Ager key definition statements are required. Key definition statements identify key data contained in the defined screen image and have the following format and content:

```
                        +--------------- key data screen row
                        |   +------------ key data screen column
                        |   |   +--------- key data length
                        |   |   |   +----- opt: Pos, Row, Col, Display
                        |   |   |   |
        | set &key to "rr  cc  lll o"
Columns: +----+----1----+----2----+----3----+----4----+----5
         0         0         0         0         0         0
```

One or more Session Ager date definition statements may be specified. Date definition statements identify date fields and the format of dates in the defined screen image that are to be aged, and have the following format and content:

```
                        +--------------- date field screen row
                        |   +------------ date field screen column
                        |   |   +--------- date field length
                        |   |   |   +----- end row for def ditto
                        |   |   |   |   +--date picture
                        |   |   |   |   |
        | set &key to "rr  cc  lll ee  month,  d yyy
Columns: +----+----1----+----2----+----3----+----4----+----5
        0         0         0         0         0         0
```

A sample Session Ager screen definition follows:

```
+Session Step(PT14V000 PF3--> CICS33S) Cursor(24,13) --------- #00022
11:34:38.33+
| JANUARY 29, 1998 PLATINUM TECHNOLOGY INC. 11:34 AM |
| EMPLOYEE INFORMATION |
| REPORT |
| |
| |
| |
| WORK BIRTH CENTURY |
| DEPT NAME DATE BIRTHDATE SALARY |
| ---- ------------------------------ ---------- ---------- -------------- |
| |
| A00 SEAN O'CONNELL 10/18/1942 10/18/2042 $29,250.00 |
| GREG ORLANDO 10/18/1942 10/18/2042 $29,250.00 |
| C01 SALLY A KWAN 05/11/1941 05/11/2041 $38,250.00 |
| KIM N NATZ 01/19/1946 01/19/2046 $28,420.00 |
| HEATHER A NICHOLLS 01/19/1946 01/19/2046 $28,420.00 |
| DOLORES M QUINTANA 09/15/1925 09/15/2025 $23,800.00 |
| D11 BRUCE ADAMSON 05/17/1947 05/17/2047 $25,280.00 |
| DAVID BROWN 05/29/1941 05/29/2041 $27,740.00 |
| REBA K JOHN 03/19/1948 03/19/2048 $29,840.00 |
| WILLIAM T JONES 02/23/1953 02/23/2053 $18,270.00 |
| JENNIFER K LUTZ 03/19/1948 03/19/2048 $29,840.00 |
| ELIZABETH R PIANKA 04/12/1955 04/12/2055 $22,250.00 |
| Page: 1 OF 1 Cols: 1 - 78 LINES: 1 - 22 |
| Next Page: PFKEYS:3=END;4=PRNT;6/9=P UP/DWN;7/8=L UP/DWN;10/11=LFT/RGT |
+-------------------------------------------------------------------------------
-+
set &key to "02 27 020 P"
set &pic to "01 03 019 01 mn d, yyyy"
set &pic to "11 40 010 22 mm/dd/yyyy"
set &pic to "11 52 010 22 mm/dd/yyyy"
```

The Trace Utility

The CA Enterprise Tester Trace Utility reads a session trace data set that was captured using the trace Format option, edits and/or reformats trace data as requested, and prints or saves the reformatted trace data. Trace Utility options may be used to add DELAY statements to a trace that was captured without delays, may be used to remove unnecessary screen images or unwanted PSL statements, or may be used to create printable screen images to cut-and-paste into application documentation.

To use the Trace Utility, enter the UTILITY command at the Control Panel and select Utility Menu option T to display the Trace Utility panel.

```
+-------------------------------------------------------------------------------+
| ProTerm Trace Utility ------------------------------------------------------- |
| COMMAND ===>                                                                  |
|                                                                               |
|      ENTER TO PROCESS                                                         |
|  PF3 END TO EXIT PANEL                                                        |
|                                                                               |
| To print or reformat a traced session, enter source Format trace data set:    |
|                                                                               |
| Data Set Name ===> PROTERM.TRACE                                              |
| Member Name   ===>           ( Blank or pattern for member selection list )   |
|                              ( Select codes: S Select, B Browse, E Edit )     |
| Enter target trace data set and member, or SYSOUT=class,DEST=destname:        |
|                                                                               |
| Data Set Name ===> PROTERM.TRACE                                              |
| Member Name   ===>           ( Blank or pattern for member selection list )   |
|                                                                               |
| Protect target? ===> YES     ( Yes or No to replace data )                    |
| Pack data?      ===> YES     ( Yes or No )                                    |
|                                                                               |
| Trace type      ===> F       ( F Format for Analyze, I Image, U Uppercase )   |
|                                                                               |
| Trace data options:          ( Yes or No )                                    |
|                                                                               |
| Delays?    ===> YES Logons?    ===> NO Free KB only?    ===> YES              |
+-------------------------------------------------------------------------------+
```

Trace Utility Services

To display a PDS or library member selection list:

1. Enter the name of a trace PDS or library as the source or target Data Set Name.
2. Enter blanks or a member name pattern as the source or target Member Name.
3. Press ENTER.

   While using a member selection list, members may be selected, browsed, or edited using S, B, or E member list select codes.

To print or reformat a Format session trace data set:

1. Enter the trace data set and member name to be processed as the source Data Set Name and Member Name.
2. To print a trace, enter a SYSOUT class and optional DEST as the target Data Set Name.
3. To save reformatted trace data in a data set, enter the target Data Set Name and Member Name. A target data set must be an existing sequential data set, PDS or library with attributes RECFM=F and LRECL=133-256.
4. Enter an appropriate option value for each Trace Utility panel option. Trace Utility options are explained below.
5. Press ENTER.

To exit the Trace Utility panel:

Enter the END command or press a PF key assigned to the END command.

Trace Utility Options

Protect Target?

Enter Yes to avoid overlaying existing target data or enter No to replace any existing target data.

Pack Data?

Enter Yes to write data to a library member using the ISPF packed data format, otherwise, enter No. When a trace data set is printed, target data is never packed.

Trace Type?

Enter the type of trace to be written, either F to trace screen images containing buffer Format information, I to trace printable virtual terminal screen Images, or U to trace Upper case printable virtual terminal screen images. When a trace is printed, Trace type F is printed as Trace type I.

Delays?

Enter Yes to trace PSL DELAY statements, otherwise enter No.

Logons?

Enter Yes to trace PSL LOGON statements, otherwise enter No.

Free KB Only?

Enter Yes to trace outbound virtual terminal screen images only when the terminal keyboard was freed by a message, or enter No to trace all source virtual terminal screen images.

The Record Utility

The CA Enterprise Tester Record Utility reads a session trace data set that was captured using the trace Format option, and records a script or REXX exec that creates a session similar to the traced session when the script or exec is executed. When a session trace is recorded as a script or REXX exec, user keystrokes and other session events captured as traced screen images are recorded as PSL statements or commands for REXX. A script or exec created by the Record Utility is similar to a script or exec that would have been created if the traced session were originally recorded as a script or exec. To use the Record Utility, enter the UTILITY command at the Control Panel and select Utility Menu option R to display the Record Utility panel.

```
+-------------------------------------------------------------------------------
-+
| ProTerm Record Utility ------------------------------------------------------
-|
| COMMAND ===> |
| |
| ENTER TO PROCESS |
| PF3 END TO EXIT PANEL |
| |
| To record a traced session in PSL or REXX, enter source Format trace data set:
|
| |
| Data Set Name ===> PROTERM.TRACE |
| Member Name ===> ( Blank or pattern for member selection list ) |
| ( Select codes: S Select, B Browse, E Edit ) |
| Enter target script or exec library and member name: |
| |
| Data Set Name ===> PROTERM.SCRIPT |
| Member Name ===> ( Blank or pattern for member selection list ) |
| |
| Protect target? ===> YES ( Yes or No to replace data ) |
| Pack data? ===> YES ( Yes or No ) |
| Screen comments ===> 03 ( Record 0 - 43 screen rows before each TYPE ) |
| Language ===> PSL ( PSL or REXX ) |
| |
| Record data options: ( Yes or No ) |
| |
| Delays? ===> YES MDT Fields? ===> NO |
+-------------------------------------------------------------------------------
-+
```

Record Utility Services

To display a PDS or library member selection list:

1. Enter the name of trace, script or exec library as the source or target Data Set Name.
2. Enter blanks or a member name pattern as the source or target Member Name
3. Press ENTER.

While using a member selection list, members may be selected, browsed, or edited using S, B, or E member list select codes.

To record a session that was previously traced using the trace Format option as a script or REXX exec:

1. Enter the trace data set and member name to be recorded as the source Data Set Name and Member Name.
2. Enter the script or exec library member name as the target Data Set Name and Member Name. The target data set must be an existing PDS or library with attributes RECFM=F and LRECL=80.
3. Enter an appropriate option value for each Record Utility panel option. Record Utility options are explained below.
4. Press ENTER.

To exit the Record Utility panel:

- Enter the END command or press a PF key assigned to the END command. Record Utility Options

Record Utility Options

Protect Target?

Enter Yes to avoid overlaying existing target data or enter No to replace any existing target data.

Comments?

Enter the number of screen rows, from 0 through 43, to be recorded as comments preceding each recorded PSL TYPE statement or command.

Pack Data?

Enter Yes to record a script using the ISPF packed data format, otherwise, enter No. Because REXX does not support packed execs, a recorded exec is never packed.

Language

Enter PSL to record a session as a script, or enter REXX to record a session as a REXX exec. When a traced session is recorded, user keystrokes and other session events represented by traced screen images are converted to PSL LOGON, TYPE, INVITE, DELAY, and LOGOFF statements or commands for REXX. A script or exec created by the Record Utility is similar to a script or exec that would have been created if the traced session were originally recorded as a script or exec, and may be executed like any other script or exec to recreate the captured session.

Delays?

Enter Yes to record PSL DELAY statements or commands for REXX, otherwise enter No.

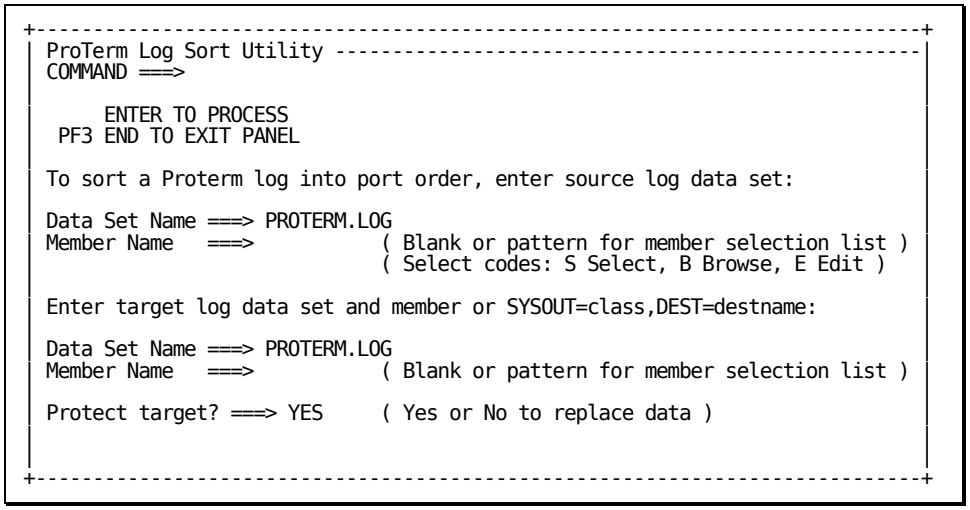
MDT Fields?

Enter Yes to record TYPE statements or commands that rekey all modifiable screen fields with a modified data tag on, or enter No to record fields only when the field value is changed or the MDT is set by user action.

The Log Sort Utility

The CA Enterprise Tester Log Sort Utility reads a log data set containing chronological messages issued by multiple scripts that were executed concurrently on multiple CA Enterprise Tester ports, for example, to drive a multi-session online system test, and re-orders the messages into groups containing all messages for each port. The Log Sort Utility may be used to print or save a re-ordered log to view all messages related to events that occurred on each port separate from messages that document events that occurred concurrently on other ports.

To use the Log Sort Utility, enter the UTILITY command at the Control Panel and select Utility Menu option L to display the Log Sort Utility panel.



Log Sort Utility Services

To display a PDS or library member selection list:

1. Enter the name of log PDS or library as the source or target Data Set Name.
2. Enter blanks or a member name pattern as the source or target Member Name.
3. Press ENTER.

While using a member selection list, members may be selected, browsed, or edited using S, B, or E member list select codes.

To sort a multi-port log data set into order by port:

1. Enter the log data set name or data set and member name to be sorted as the source log Data Set Name and Member Name.
2. To print a re-ordered log, enter a SYSOUT class and optional DEST as the target Data Set Name.

To save a re-ordered log in a data set, enter the target Data Set Name and Member Name:

A target data set must be an existing sequential data set, PDS or library with attributes RECFM=F and LRECL=151.

1. If a re-ordered log is written to a target data set or member, enter Yes as the ProTect Target? option value to avoid overlaying existing target data, or enter No to replace any existing target data.
2. Press ENTER.

To exit the Log Sort Utility panel:

Enter the END command or press a PF key assigned to the END command.

The Batch Play Utility

The CA Enterprise Tester Batch Play Utility sets up a job that may be submitted to play a script in batch.

To use the Batch Play Utility, enter the UTILITY command at the Control Panel and select Utility Menu option P to display the Batch Play Utility panel.

```
+-------------------------------------------------------------------------------+
| ProTerm Batch Play Utility ----------------------------------------------------|
| COMMAND ===>                                                                  |
|      ENTER TO PROCESS                                                         |
|  PF3 END TO EXIT PANEL                                                        |
| To set up a job to play a script in batch, enter script to be played:         |
|                                                                               |
| Library Name ===> PROTERM.SCRIPT                                              |
| Script Name  ===>            ( Blank or pattern for member selection list )   |
|                              ( Select codes: S Select, B Browse, E Edit )     |
| EXEC PARM    ===>                                                             |
|                              ( ie., FTRACE,var1=value1,var2=value2 )          |
| Enter any other that contain linked-to scripts:                               |
| Library 1    ===>                                                             |
| Library 2    ===>                                                             |
| Library 3    ===>                                                             |
| To trace a batch session, enter target data set or SYSOUT=class,DEST=name:    |
| Data Set Name ===> PROTERM.TRACE                                              |
| Member Name ===> ( Blank or pattern for member selection list )               |
+-------------------------------------------------------------------------------+
```

Batch Play Utility Services

To display a script or trace library member selection list:

1. Enter the name of a PDS or library as the script Library Name or target trace Data Set Name.
2. Enter blanks or a member name pattern as the Script Name or target trace Member Name.
3. Press ENTER.

While using a member selection list, members may be selected, browsed, or edited using S, B, or E member list select codes.

To set up a job to play a script in batch:

1. Enter the script library and script to be played as the script Library Name and Script Name.
2. Enter any valid Script Language Processor keyword parameters or script variable field value assignments to be passed to the played script as the EXEC PARM value. The EXEC PARM value is set up as the JCL EXEC statement PARM value. The value must include only valid Script Language Processor keyword parameters and variable field value assignments formatted as explained in "Batch JCL" in topic 5.1.
3. If the played script links to scripts that are in script libraries other than the entered Library Name, then enter any other needed script libraries as Library 1, Library 2, or Library 3.
4. To trace a session driven by the played script, enter the target trace data set.
   - To print a trace, enter a SYSOUT class and optional DEST as the target trace Data Set Name.
   - To save traced session data in a data set, enter the target Data Set Name and Member Name. A target data set must be an existing sequential data set, PDS or library with attributes RECFM=F and LRECL=133-256. The format of the trace may be specified by entering one of the Script Language Processor keyword parameters, FTRACE or ITRACE as an EXEC PARM.
5. Press ENTER.

When valid Batch Play Utility information is entered, the Job Setup Panel is displayed. The Job Setup Panel is used to enter job statement information, a target script log data set, and to save the job in a specified JCL library. Refer to The CA Enterprise Tester Job Setup Panel for a complete description of Job Setup Panel services.

To exit the Batch Play Utility panel:

- Enter the END command or press a PF key assigned to the END command.

A Batch Play Utility Job

The Batch Play Utility uses information entered using the Batch Play Utility and Job Setup Panels to create a job that may be submitted to play a script in batch. For example:

```
//JOBNAME  JOB (ACCOUNT),'NAME',NOTIFY=UID1
//*
//* SAMPLE PROTERM BATCH JOB
//*
//PLAYSAMP EXEC PGM=PTEBS,PARM='FTRACE,USERID=UIDB,TOKEN=UID1B'
//PTELIB   DD DISP=SHR,DSN=SITE.PROTERM.SCRIPT
//    DD DISP=SHR,DSN=GROUP.PROTERM.SCRIPT
//PTETRACE DD DISP=SHR,DSN=UID1.PROTERM.TRACE(PLAYSAMP)
//PTELOG   DD DISP=SHR,DSN=UID1.PROTERM.LOG(PLAYSAMP)
//PTEIN    DD DISP=SHR,DSN=UID1.PROTERM.SCRIPT(PLAYME)
```

For complete information about the job control statements, data sets, and script parameters used to play a script in batch, refer to Batch JCL.

The Multi-Port Select Utility

The Multi-Port Select Utility creates select scripts, that may be played on multiple ports using the Play Panel Number of Ports option, to select and link to a different script, and to optionally select user-defined data, for each port on which it is played. Additional information about select scripts may be found in Using Select Scripts to Manage Multi-Session Tests.

To use the Multi-Port Select Utility, enter the UTILITY command at the Control Panel and select Utility Menu option S to display the Multi-Port Select Utility panel.

```
+-------------------------------------------------------------------------------+
| ProTerm Multi-Port Select Utility ------------------------------------------- |
| COMMAND ===>                                                                  |
|     ENTER TO PROCESS                                                          |
| PF3  END TO EXIT PANEL                                                        |
|To set up a Select Script enter pattern Link Script name to select on TEPORT: |
|                                                                               |
| Logon Script  ===>   ( Optional script linked to first on all ports )         |
| Link Script   ===>   ( In pattern, * is port 1-999, *** is 001-999 )          |
| Logoff Script ===>   ( Optional script linked to last on all ports )          |
|                                                                               |
| To set up logic to select data for PTEPORT, enter:                            |
| Field Names   ===>                                                            |
| Data Patterns ===>                                                            |
| Table Entries ===> 0        ( 0 to set up logic to generate data for port )   |
|                             ( 1 - 999 to set up and select port table data )  |
| Enter script library and target Select Script name:                           |
|                                                                               |
| Library Name  ===> PROTERM.SCRIPT                                             |
| Select Script ===>          ( Blank or pattern for member selection list )    |
|                             ( Select codes: S Select, B Browse, E Edit )      |
| Protect target?  ===> YES   ( Yes or No to replace data )                     |
+-------------------------------------------------------------------------------+
```

Multi-Port Select Utility Services

To display a script library member selection list:

1. Enter the name of a script library as the script Library Name.
2. Enter blanks or a member name pattern as the target Select Script name.
3. Press ENTER.

While using a member selection list, members may be selected, browsed, or edited using S, B, or E member list select codes.

To set up a Select Script:

1. Optionally, enter the name of a Logon Script to be linked to on all ports, after data has been selected for a port, and before linking to the script selected for the port. A Logon Script may be used to start a session and to sign on to ready a session for a Link Script that enters only application transactions, and may use data selected by a Select Script.

   For example, user ID and token may be defined as Field Names to create a Select Script that selects a user ID and password token for a port, and a general purpose Logon Script may enter selected user IDs and password tokens by typing PSL variables, &user ID and &token.

2. Enter a pattern Link Script name, as follows:

   If the names of scripts to be selected for the current value of PSL variable, &PTEPORT, have a common constant prefix and a port number as a suffix, enter a pattern Link Script name that begins with the common constant prefix and ends with a string of one or more asterisks that represent a port number.

   For example, if scripts named SESS1 through SESS10 are to be linked to on ports 1 through 10, enter SESS* as the pattern Link Script name. If scripts named SESS001 through SESS100 are to be linked to on ports 1 through 100, enter SESS*** as the pattern Link Script name.

   If the names of scripts to be selected for &PTEPORT do not conform to a constant-port naming convention, enter any pattern name, for example, PORT*, and enter the number of scripts to be played as the value of Table Ports. When the select script is subsequently displayed using ISPF EDIT, replace the pattern names in the &port-data table created by the Select Utility, with the names of the scripts to be selected for each port.

3. Optionally, enter a the name of a Logoff Script to be linked to on all ports, after linking to the script selected for the current port. A Logoff Script may be used to end a session started by a Logon Script or to perform any other desired function before a Select Script ends.

4. To set up logic to select data for &PTEPORT, enter a list of one or more PSL variable Field Names whose value is to be selected by the Select Script, and enter corresponding Data Patterns for all entered Field Names. Enter Field Names without the leading ampersand used to refer to PSL variables in a script. Enter a blank between all entered Field Names and Data Patterns.

   Data Patterns may be any string of data and may include a string of asterisks, that refer to the value of &PTEPORT in the same way that pattern Link Script names refer to the value of &PTEPORT. The length of the data set up for entered Field Names is determined by corresponding Data Patterns. A single asterisk in a data pattern reserves space for three characters for the maximum value of &PTEPORT.

   Like pattern Link Script names, when data values to be selected do not conform to a constant-port value convention, enter the number of Table Ports to be set up, and edit the &port-data table set up by the Select Utility when the Select Script is displayed using ISPF EDIT, replacing pattern data values with the real values to be selected for each port.

5. If Link Script names and the values of all entered Field Names to be selected by a Select Script conform to a constant-port value, enter 0 Table Entries, to create a script that generates a Link Script name and a value for all entered Field Names using the current value of &PTEPORT and the entered pattern Link Script name and Data Patterns.

   To set up a Select Script that creates a &port-data table that contains a Link Script name and values for all Field Names for each port, enter the number of &port-data Table Entries to be set up. The number of entries in a &port-data table and the values in each entry may be changed at any time using ISPF EDIT.

6. Enter the script Library Name that contains all entered Logon Script, Link Script, and Logoff Script names, and to which the Select Script created by the Select Utility is to be written. The Library Name specified must be an existing PDS or library with attributes RECFM=F and LRECL=80.

7. Enter the name of the Select Script to be created by the Select Utility.

8. Enter Yes as the ProTect Target?option value to avoid overlaying existing target data, or enter No to replace any existing target data.

9. Press ENTER.

   When valid Multi-Port Select Utility information is entered, a select script is created and saved in the specified Select Script member name and Library Name, and the select script is displayed using ISPF EDIT, allowing pattern data in a &port-data table to be immediately changed to real script names and/or data values to be selected.

To exit the Multi-Port Select Utility panel:

- Enter the END command or press a PF key assigned to the END command.

Multi-Port Select Scripts

The Multi-Port Select Utility may be used to create several types of select scripts. The following examples illustrate some of the possibilities.

- A number of complete sessions may be recorded using script names that begin with a constant, for example 'PORT', and that end with the number of the port on which each script is to be played when the scripts are used to drive a multi-session system test.

  The Multi-Port Select Utility may be used to create a select script to be played on multiple ports, that generates the name of and links to the script recorded for each port. The following script was created by entering PORT* as the pattern Link Script name, and a target script library and Select Script name.

  ```
  set &script to 'PORT' &pteport
  log 'Port ' &pteport ' linking to ' &script
  link &script
  log 'Port ' &pteport ' link    to ' &script ' complete'
  ```

  The resulting script selects the name of a script for each port on which it is played by concatenating the constant value, 'PORT', with the number of the port on which it is executing, &pteport, and links to the selected script. The Select Utility always creates PSL LOG statements that document the script selected for each port, and when the script started and ended.

- The Multi-Port Select Utility also sets up logic to select values for user-defined fields, for each port. For example, a select script may be set up that selects a user ID and password to be used to start a session on each port, and the user ID and token selected may be used by a standard logon script to start sessions and sign on, before the select script links to scripts selected for each port, that enter only application transactions on the ready sessions.

  For example, if the scripts in the previous example were created by starting sessions and signing on before starting a session recording, a select script may be created that starts a session and signs on using a selected user ID and password token, before linking to a selected script.

  The select script shown below was set up to manage a set of test user IDs, that begin with the constant 'TEST' and end with a three digit sequence number, and to use a single password token that represents the a password that is used for all test user IDs. A script named STDLOGON uses the selected user ID and password token to start a session and sign on. The following information was entered at the Multi-Port Select Utility panel:

  ```
  Logon Script    ===> STDLOGON
  Link Script     ===> PORT*
  Field Names     ===> USERID  TOKEN
  Data Patterns   ===> test*** test1a
  Table Entries   ===> 0
  ```

The information was used by the Multi-Port Select Utility to create the following select script.

```
set &script to 'PORT' &pteport
set &USERID to 'test***'
set &USERID(,5,3) to zoned &pteport
set &TOKEN  to 'test1a'
log 'Port ' &pteport ' USERID = ' &USERID
log 'Port ' &pteport ' TOKEN  = ' &TOKEN
link 'STDLOGON'log 'Port ' &pteport ' linking to ' &script
link &scriptlog    'Port ' &pteport ' link to    ' &script '
complete'
```

- The select scripts in the previous examples are simple, efficient, and may be played on any number of ports for which scripts, user IDs or other data have been set up. However, they are useful only when script names and field values are constant or have a constant-port pattern. Select Utility panel Table Entries may be specified to set up another type of select script that creates and selects script names and field values from a table, using &PTEPORT to locate the table entry that contains the script name and field values for each port.

  The following select script was created by entering the same information used in the previous example, except that 10 Table Entries was specified.

```
set &port-data to
  test001test1a'
   'PORT2
  test002test1a'
   'PORT3
  test003test1a'
   'PORT4
  test004test1a'
   'PORT5
  test005test1a'
   'PORT6
  test006test1a'
   'PORT7
  test007test1a'
   'PORT8
  test008test1a'
   'PORT9
  test009test1a'
   'PORT10
  test010test1a'
compute &port-p = &pteport * 21 - 20
   set &port-data to &port-data(,&port-p,21)
   set &script    to &port-data(,1,8)
   set &user ID    to &port-data(,9,7)
   set &token to &port-data(,16,6)
log
'Port ' &pteport ' user ID = ' &useridlog
```

```
'Port ' &pteport ' token  = ' &tokenlink 'STDLOGON'log
'Port ' &pteport ' linking to ' &scriptlink &scriptlog
'Port ' &pteport ' link to ' &script ' complete'
```

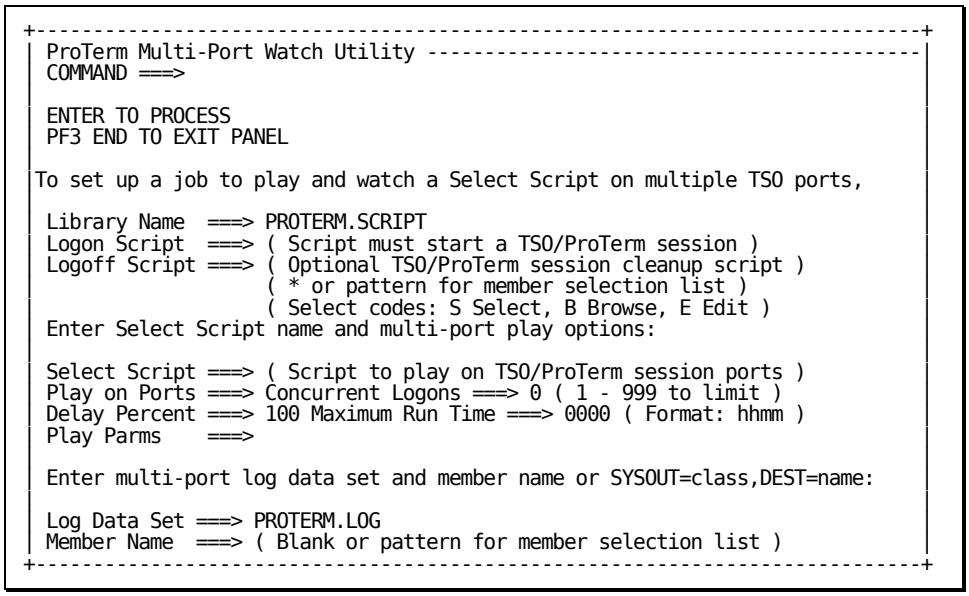
The &port-datatable created when Table Entries are specified contains an entry for each specified port. The table entry for each port contains the script name and values of defined fields to be selected for the port.

The Select Utility uses the constant or pattern values entered as a Link Scriptor as Data Patterns to create initial &port-data table names and values for each port. The script names and field values in a port-data table may be changed to any desired values using ISPF EDIT. A select script that uses a &port-data table may be easily modified to be played on a larger number of ports, by adding entries to the &port-data table to specify the scripts and data to be selected for the additional ports.

The Multi-Port Watch Utility

The CA Enterprise Tester Multi-Port Watch Utility creates a job that may be submitted to play a select script on multiple ports, and to watch and otherwise manage the played scripts. A job created by the Watch Utility executes a distributed script, PTEMPLAY, that uses a virtual terminal session with TSO to access the Interactive System, and uses the Play Panel and other Interactive System functions in the same way that a person would, to play scripts, to watch script activity, to detect and document problems, and to perform other functions. The Multi-Port Watch Utility and the PTEMPLAY script are designed to play scripts like the select scripts created by the Multi-Port Select Utility.

To use the Multi-Port Watch Utility, enter the UTILITY command at the Control Panel and select Utility Menu option W to display the Multi-Port Watch Utility Panel.



Multi-Port Watch Utility Services

To display a script library member selection list:

1. Enter the name of a script library as the script Library Name.
2. Enter blanks or a member name pattern as the Logon Script or Select Script name, or enter a member name pattern as the Logoff Script name.
3. Press ENTER. While using a member selection list, members may be selected, browsed, or edited using S, B, or E member list select codes.

To set up a job to play and watch a Select Script on multiple TSO/CA Enterprise Tester ports:

1. Enter the script Library Name that contains all scripts entered as a Logon Script, Logoff Script, or Select Script.
2. Enter the name of a Logon Script that starts a TSO session and that uses the session to display the Control Panel. A subsequent topic, Watch Job User IDs and TSO Logon Scripts , explains important information about the user ID used to start a TSO session for use by a watch job, and a provides a procedure that may be used to create a suitable Logon Script.

3. Optionally, enter the name of a Logoff Script to be linked to after all played scripts running on TSO/CA Enterprise Tester session ports have completed or are stopped. A Logoff Script may be used to insure orderly termination of the TSO session used by the Multi-Port Watch job. If a Logoff Script is not specified or a specified script does not end the TSO session, PTEMPLAY enters =x and logoff to end the TSO session, and if necessary, issues a PSL LOGOFF statement to end the TSO session.
4. Enter the name of a single script to be played on multiple TSO/CA Enterprise Tester ports as the Select Script. A select script like the scripts created by the Multi-Port Select Utility, may be played to select and link to different scripts on each port on which the Select Script is played.
5. Enter the number of ports, 1to 999, on which the Select Script is to be played, as the value of Play on Ports.
6. If the scripts to be played or linked to by the Select Script contain PSL DELAY statements, enter a Delay Percent, from 0to 999, to specify the percent of DELAY statement times to be used. Delay Percent regulates the speed of played scripts by decreasing or increasing the think time delays that occur when DELAY statements are executed.
7. Enter any Play Parms to be entered at the Play Panel when Select Script is played. For a complete description of valid parameter values, refer to Batch JCL.
8. Enter the maximum number of Concurrent Logons, 0to 999, issued by played or linked-to scripts that may be active at the same time. 0 (zero) allows any number of logons to be concurrently active. The option may be used to prevent a VTAM SOS condition, that could occur if the number of sessions being started at the same time, consumes VTAM buffers, that were tuned to support a smaller number of concurrent logons.
9. To limit the elapsed time that played scripts are permitted to run, enter a Maximum Run Time, expressed in the form, hhmm, where hh is hours and mm is minutes. If 0 or no time is entered, no time limit is imposed. When a specified run time is exceeded, PTEMPLAY stops all played scripts that are still active, ends all active scripts and sessions, ends the watch job TSO session, documents the results and ends.
10. Enter a multi-port to be used to capture messages logged by all played scripts. Messages from all ports are written to the multi-port log data set chronologically.

    To print logged messages, enter a SYSOUT class and optional DEST as the Log Data Set.

    To save logged messages in a data set, enter the Log Data Set and Member Name. The log data set must be an existing sequential data set, PDS or library with attributes RECFM=F and LRECL=151. The Log Sort Utility may be used to sort messages in a multi-port log data set into port order.
11. Press ENTER.

When valid Multi-Port Watch Utility information is entered, the Job Setup Panel is displayed. The Job Setup Panel is used to enter job statement information, a target script log data set used to capture messages logged by the PTEMPLAY script, and to save the job in a specified JCL library. Refer to The CA Enterprise Tester Job Setup Panel, for a complete description of Job Setup Panel services.

To exit the Multi-Port Watch Utility Panel:

- Enter the END command or press a PF key assigned to the END command.

A Multi-Port Watch Utility Job

The Multi-Port Watch Utility uses information entered using the Multi-Port Watch Utility and Job Setup Panels, to create a job, like the job shown in the following example, that may be scheduled to be executed during non-business hours to conduct a multi-session online system test automatically.

```
//JOBNAME JOB (ACCOUNT),'NAME',NOTIFY=UID1
//*
//* SAMPLE MULTI_PORT WATCH JOB
//*
//WATCH EXEC PGM=PTEBS
//PTELIB DD DISP=SHR,DSN=UID1.PROTERM.SCRIPT
//PTETRACE DD SYSOUT=*
//PTELOG DD SYSOUT=*|
//PTEIN DD *
set &library to "'UID1.PROTERM.SCRIPT'"
set &logon to 'TSOLOGON'
set &logoff to ''
set &select to 'MPSELECT'
set &ports to 35
set &delay to 0
set &think to '0000'
set &parms to 'RATE=25,PORTS=35'
set &logons to 0
set &runtime to '0100'
set &review to 4
set &logdsn to "'UID1.PROTERM.LOG'"
set &logmem to 'MULTILOG'
link 'ptemplay'
/*
```

The Multi-Port Watch Utility creates JCL required to play CA Enterprise Tester scripts in batch, and a script that sets parameter values and links to PTEMPLAY to perform watch job functions. The JCL includes a PTELIB DD statement that defines the Watch Utility panel script Library Name to the watch job, and a PTELOG DD statements that define the Job Setup Panel Log Data Set to which PTEMPLAY messages are to be logged. A PTETRACE DD statement is created to capture the TSO session used by the watch job.

The script created by the Watch Utility, following the PTEIN DD statement, includes PSL SET statements that provide information entered using the Watch Utility Panel, and other default values, when the script links to the PTEMPLAY script. The distributed version of PTEMPLAY uses the information to perform the following functions:

- PTEMPLAY links to the &logon script, to start a TSO session and to access the CA Enterprise Tester Interactive System, and uses the session to perform watch job functions.
- PTEMPLAY uses Options Panels to set Interactive System options required to support PTEMPLAY logic, and to set the number of session screen images to be saved for review, to diagnose possible session problems. The default value assigned to &review by the Multi-Port Watch Utility, may be changed to a choose another value.
- PTEMPLAY starts a log to capture messages from scripts that are subsequently played on multiple Interactive System ports, by entering the specified &logdsn and &logmem, at the Log Panel.
- PTEMPLAY uses the Play Panel to play the specified &select script on the specified number of &ports. Specified &delay percent and EXEC &parms, and a default &think time set by the Watch Utility, are entered at the Play Panel to regulate the speed of or to provide information to the &select script.
- When a value greater than 0 is specified for concurrent &logons, PTEMPLAY limits the number of scripts that are played at one time to the specified number of &logons, and plays additional scripts, up to the specified number of &ports, only when it sees that sessions started by previously played scripts are now active. The option may be used to avoid VTAM SOS or other stress related problems that may occur if an unrealistic number of sessions are started at the same time.
- After playing the &selectscript on the specified number of &ports, PTEMPLAY uses the Control Panel to watch script activity, and uses other CA Enterprise Tester panels and commands to document problems and manage scripts.
- If a problem occurs and a played script stops at an error, PTEMPLAY displays the Monitor Panel for the port and reviews and previous session screens on the port, to capture diagnostic information in the watch jobs PTETRACE data set.

- Normally, PTEMPLAY ends when all played scripts have ended. But if a &runtime limit is specified and all scripts have not ended within the specified limit, PTEMPLAY interrupts and terminates all active scripts and sessions.
- When all played scripts have ended or have been terminated, PTEMPLAY ends the multi-port log, restores all CA Enterprise Tester panel values altered by PTEMPLAY, and calls any specified &logoffscript, to end the watch job TSO session.

The distributed version of PTEMPLAY is just a sample. Other customized scripts that perform similar functions may be used instead of PTEMPLAY, to play and watch scripts that conduct unattended multi-session system tests. A listing of the distributed version of PTEMPLAY may be found in Batch Multi-Port Applications, and may be used as a starting point to create scripts that perform other desired functions.

Refer to A Basic Multiple Session Test for an overview of CA Enterprise Tester functions that may be used to perform multi-session tests, and for examples and suggestions that explain how to create and use multi-session test materials, including select scripts.

For complete information on JCL statements, script parameters and data sets used to play CA Enterprise Tester scripts in batch, refer to Batch JCL.

Watch Job User IDs and TSO Logon Scripts

To use the Multi-Port Watch Utility, a script must be created that starts a TSO session and that selects CA Enterprise Tester from ISPF, to display the Control Panel. The user ID used by the TSO logon script must be available to the watch job when the job is executed, and must be authorized to access data sets and other resources used within the TSO address space.

For example, the select script played by a watch job and all scripts linked to by the select script run in the TSO address space. If the select script or linked to scripts use password tokens to start sessions, password tokens must be set up for the user ID used for the watch job TSO session. To set up any password tokens to be used by the user ID, use the user ID to start a TSO session, and use the CA Enterprise Tester Token Security panel to display a token screen named the same as each token, from anyone's CA Enterprise Tester TOKEN data set, and enter the correct password for each token, as explained in Token Security.

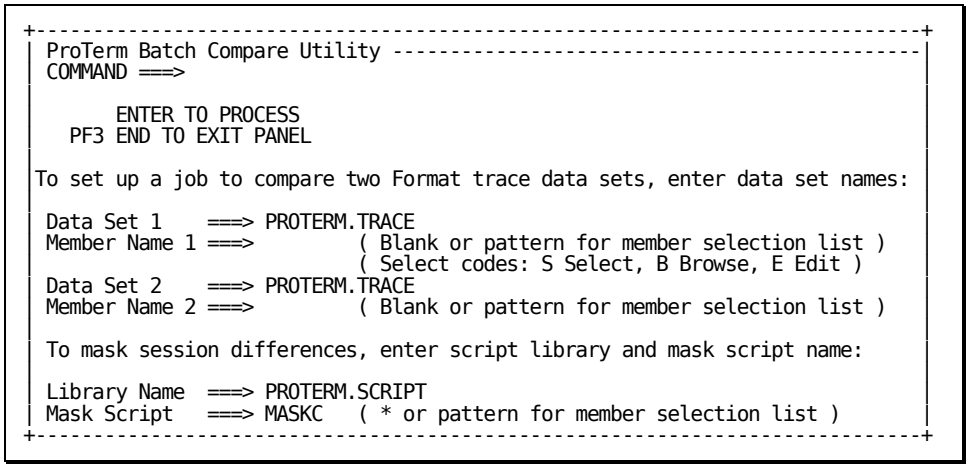
To Create a Suitable Logon Script:

1. Using the Interactive System RECORD command, start a session recording to record a subsequent session as a CA Enterprise Tester script. Specify PSL as the language.
2. Using the Interactive System SESSION command, start a session with TSO, on the port being recorded. Sign on using a TSO user ID to be used by the watch job to start a TSO session.
3. Enter the TSO or ISPF command, CLIST, REXX exec, or ISPF menu selection that is used to select CA Enterprise Tester and to display the Control Panel.
4. End the recording of the TSO session by first pressing a PF key that is assigned to the ZOOM or END command or press the PA2 key to display a CA Enterprise Tester command field, and then end the recording by entering the END RECORD command.
5. End the TSO session that was recorded.
6. If it is sometimes necessary to press enter multiple times to get past TSO logon messages or other TSO session startup activity. You may wish to add logic to the recorded script to make it reliably get to the CA Enterprise Tester Control Panel.

The Batch Compare Utility

The CA Enterprise Tester Batch Compare Utility sets up a batch job that may be submitted to compare two session trace data sets that were previously captured using the trace Format option.

To use the Batch Compare Utility, enter the UTILITY command at the Control Panel and select Utility Menu option C to display the Batch Compare Utility Panel.



Batch Compare Utility Services

To display a trace or script library member selection list:

1. Enter the name of a trace PDS or library as Data Set1or Data Set 2, or enter a script Library Name.
2. Enter blanks or a member name pattern as Member Name 1 or Member Name 2, or enter a member name pattern as the Mask Script name.
3. Press ENTER.

   While using a member selection list, members may be selected, browsed, or edited using S, B, or E member list select codes.

To set up a job to compare two session trace data sets:

1. Enter the first data set as Data Set 1 and Member Name 1.
2. Enter the second data set as Data Set 2and Member Name 2.
3. To mask session differences, enter the script Library Name and Mask Script to be linked to by the trace compare script, to select screen fields to be masked. For information about mask scripts, refer to Masking Dates, Times, and Other Data.
4. Press ENTER.

When valid Batch Compare Utility information is entered, the Job Setup Panel is displayed. The Job Setup Panel is used to enter job statement information, a target script log data set, and to save the job in a specified JCL library. Refer to The CA Enterprise Tester Job Setup Panel for a complete description of Job Setup Panel services.

To exit the Batch Compare Utility Panel:

- Enter the END command or press a PF key assigned to the END command.

A Batch Compare Utility Job

The Batch Compare Utility uses information entered using the Batch Compare Utility and Job Setup Panels to create a job that may be submitted to compare two previously traced sessions that were captured using the trace Format option. Jobs created by the trace utility may be used to compare online system responses to previous online system responses received when tests were previously performed, to identify differences, when scripts are used to repeat the tests automatically.

A job created by the Batch Compare Utility, uses a distributed CA Enterprise Tester script, PTECTTB to compare the traced sessions.

```
//JOBNAME  JOB (ACCOUNT),'NAME',NOTIFY=UID1
//*
//* SAMPLE PROTERM SESSION TRACE COMPARE JOB
//*
//COMPARE  EXEC PGM=PTEBS
//TRACE1 DD DISP=SHR,DSN=UID.PROTERM.TRACE(SESSION1)
//TRACE2 DD DISP=SHR,DSN=UID.PROTERM.TRACE(SESSION2
//PTELIB DD DISP=SHR,DSN=UID.PROTERM.SCRIPT
//PTELOG DD DISP=SHR,DSN=UID.PROTERM.LOG(RESULTS)
//PTEIN  DD *
   set &mask  to 'DATEMASK'
   set &attr  to 'Y'
   set &image to 'N'
   link 'ptecttb'
/*
```

PTECTTB compares corresponding screen images found in the trace data sets specified by the TRACE1and TRACE2 DD statements. When unmasked differences are found between a pair of screen images that represent online system responses, mismatch messages similar to those produced by CA Enterprise Tester session analyzers, including full screen images of the TRACE1and TRACE2 screens that did not match, are written to the script log data set identified by the PTELOG DD statement.

A JCL parameter, MASK=script, or a SET statement that sets the script variable, &MASK, to the name of a script may be used to specify a script that masks selected session differences as described in Masking Dates, Times, and Other Data. However, because PTECTTB does not create a live session, but instead processes session trace data sets using the PSL TRACK statement, the mask script may refer only to TRACKED screen fields, when making decisions about the screen fields to be masked. If a mask script is specified, a PTELIBDD statement must be included in the JCL to identify the script library that contains the mask script.

A JCL parameter, ATTR=Y|N, or a SET statement that sets script variable, &ATTR, to Y or N, may be used to specify that screen attributes are or are not to be compared. By default, screen attributes are compared.

A JCL parameter, IMAGE=Y|N, or a SET statement that sets script variable, &IMAGE, to Y or N, may be used to specify that screen images are to be compared (IMAGE=Y), or that the actual screen data is to be compared (IMAGE=N). When screen images are compared, attributes and nulls, which appear as blanks on a real terminal screen are converted to blanks before comparing corresponding screen data. By default, actual screen data is compared (IMAGE=N).

The CA Enterprise Tester Job Setup Panel

The CA Enterprise Tester Job Setup Panel is used to set up and save a job that executes a CA Enterprise Tester script in batch. The Job Setup Panel is displayed when valid job information is entered using the Batch Play Utility, Batch Compare Utility or the Multi-Port Watch Utility.

```
+-------------------------------------------------------------------------------+
| ProTerm Job Setup Panel ------------------------------------------------------|
| COMMAND ===>                                                                  |
|                                                                               |
|      ENTER TO PROCESS                                                         |
|  PF3 END TO EXIT PANEL                                                        |
|                                                                               |
| Enter job statement information:                                              |
|                                                                               |
| //JOBNAME JOB (ACCOUNT),'NAME',                                               |
| // CLASS=A,MSGCLASS=A,NOTIFY=UID                                              |
| //*                                                                           |
| //*                                                                           |
|                                                                               |
| Enter ProTerm log data set and member, or SYSOUT=class,DEST=name:             |
|                                                                               |
| Log Data Set    ===> PROTERM.LOG                                              |
| Member Name     ===>      ( Blank or pattern for member selection list )      |
|                           ( Select codes: S Select, B Browse, E Edit )        |
| Enter target JCL library and member name:                                     |
|                                                                               |
| JCL Library     ===>                                                          |
| Member Name     ===>      ( Blank or pattern for member selection list )      |
|                                                                               |
| Protect target? ===> YES  ( Yes or No to replace data )                       |
+-------------------------------------------------------------------------------+
```

Job Setup Services

To display a log or JCL library member selection list:

1. Enter the name of a log or JCL PDS or library as the Log Data Set or JCL Library.
2. Enter blanks or a member name pattern as the log or JCL library Member Name.
3. Press enter.

While using a member selection list, members may be selected, browsed, or edited using S, B, or E member list select codes.

To set up and save a job:

1. Enter job statement information. Up to four lines of information may be entered including a JOB statement, //*JCL comments, JES /* statements, a JOBLIBDD statement, or any other valid JCL that may proceed a JCL EXEC statement.
2. Enter a CA Enterprise Tester log data set.

   To print logged messages, enter a SYSOUT class and optional DEST as the Log Data Set.

   To save logged messages in a data set, enter the Log Data Set and Member Name. A target data set must be an existing sequential data set, PDS or library with attributes RECFM=F and LRECL=151.
3. Enter the target JCL Library and Member Name to which the job is to be written. The JCL Library specified must be an existing PDS or library with attributes RECFM=F and LRECL=80.
4. Enter Yes as the ProTect Target?option value to avoid overlaying existing target data, or enter No to replace any existing target data.
5. Press ENTER.

Valid information entered using the Batch Setup Panel and information entered using a previous job setup utility panel is combined and a job is written to the target JCL Library and Member Name specified.

After writing a job to a JCL library member, the library member is automatically displayed using ISPF EDIT, allowing the JCL and PSL control statements created to be modified and saved or to be submitted immediately using the SUBMIT command.

To exit the Job Setup Panel:

- Enter the END command or press a PF key assigned to the END command.

The CA Enterprise Tester Command Monitor for REXX

The CA Enterprise Tester Command Monitor For REXX is an interactive utility that may be used to execute and monitor a recorded or user-written REXX exec that uses CA Enterprise Tester session commands, to display error messages, commands, exec-driven session events and other information at a terminal, while the exec is executing.

The Command Monitor may also be used to manage a system stress test or other multi-session online system test that is driven by a list of execs that create CA Enterprise Tester sessions. When a list of execs is specified, the Command Monitor attaches each exec as an asynchronous z/OS task, and displays virtual terminal screen images and other events driven by the attached execs in windows, allowing the user to view multiple concurrent session events at a single terminal at the same time, while the test is in progress.

Refer to CA Enterprise Tester Commands for REXX for complete information about the Command Monitor panel, fields, options and functions.

Chapter 10: Commands for REXX

This chapter describes the CA Enterprise Tester Host Command Environment which allows REXX execs to ADDRESS PROTERM Tester and use virtual terminals in a multi-session synchronous or asynchronous environment.

- CA Enterprise Tester Commands for REXX
- The Session Commands
- Tracing Exec-Driven Sessions
- CA Enterprise Tester Ports and Multiple Sessions
- Sharing Sessions with Called Execs
- Attaching Asynchronous Exec Tasks
- Queues and Intertask Communication
- Sharing Resources with Attached Exec
- Smart Exec-Driven Sessions
- ISPF Session Panels
- Monitoring Execs and Sessions
- The CA Enterprise Tester Command Monitor for REXX

The CA Enterprise Tester Host Command Environment allows REXX execs to ADDRESS PROTERM Tester and to issue CA Enterprise Tester host commands.

- CA Enterprise Tester LOGON, TYPE, INVITE, DELAY, and LOGOFF commands allow REXX execs to start sessions using virtual terminals and to use online software that was designed for interactive use by people using 3270 terminals. Using CA Enterprise Tester Interactive System functions, REXX execs that repeat captured sessions may be created automatically by recording interactive user sessions or by converting previously traced sessions to REXX execs.
- CA Enterprise Tester TRACE and ENDTRACE commands allow execs to capture exec-driven sessions in datasets. Using CA Enterprise Tester Interactive System functions, traced virtual terminal screen images may be displayed, printed, or used to perform regression testing, or may be converted to CA Enterprise Tester scripts or REXX execs that recreate captured sessions when executed.
- The ATTACH command creates new z/OS tasks and REXX Language Processor Environments and uses the environments to execute REXX execs that run at the same time as the attaching exec. Attached exec tasks can start and drive asynchronous sessions to conduct stress tests or to multiply throughput in production applications.

- The ACCESS command allows an exec to use multiple sessions, to attach multiple exec tasks, and to use multiples of any CA Enterprise Tester resource. Automatic resource locks, LOCK and UNLOCK commands allow multiple execs to access and share any CA Enterprise Tester resource.
- QUEUE, PUSH and POST commands allow execs to store data in designated queues to communicate with any number of other asynchronous exec tasks that process queued data using PULL or QUERY QUEUE commands.
- QUERY, HANDLE, WAIT and SIGNAL commands allow execs to inquire about CA Enterprise Tester resources, to modify or interrupt command processing, and to process errors and other conditions using REXX condition traps.
- The MONITOR command activates integrated monitor functions that display CA Enterprise Tester commands, command functions, exec-driven sessions, and error messages automatically, and may be used to debug execs or to display multiple virtual terminal screens at a single terminal to monitor concurrent session activity driven by asynchronous attached execs while performing a system stress test.

The Session Commands

CA Enterprise Tester session commands may be used by REXX execs to start sessions with online systems using virtual terminals and to use virtual terminals in the same way that people use real terminals to access online system software.

- The LOGON command allocates a virtual terminal and uses the terminal to start a session with any online system that supports terminal sessions.
- The TYPE command uses the virtual terminal keyboard to key in data, position the cursor, and to send data by pressing the Enter key, PF keys or other keys that send data.
- The INVITE command waits for an online system to send anticipated messages, like sign on prompts at the start of a session.
- The DELAY command simulates human think time and controls the speed of an exec driven session.
- The LOGOFF ends a session and frees a virtual terminal.

CA Enterprise Tester session commands return information to execs in REXX variables. At the completion of each session command, a copy of the virtual terminal display buffer is returned in a REXX variable, PTEBUFF, the virtual terminal screen size and cursor position are stored in PTEROWS, PTECOLS and PTECSRP, and other information about the virtual terminal and session is returned in other REXX variables. Using the returned display buffer, execs may analyze transaction responses and gather displayed information or use it to enter subsequent transactions.

REXX execs that repeat captured sessions may be created automatically by recording CA Enterprise Tester Interactive System sessions or by converting previously traced sessions to execs. When an exec is recorded, session initiation is captured as a LOGON command and user keystrokes are captured as TYPE commands. Recorded execs also include INVITE commands that wait for anticipated system messages and may include DELAY commands that capture actual human think-time delays that occur while recording a session. A recorded exec may be executed to repeat a captured session automatically.

By modifying a captured exec or by writing execs from scratch, intelligent REXX execs may be developed that use CA Enterprise Tester sessions and any available online system software to reliably perform or automate almost any imaginable terminal task that could by performed by a person using a terminal.

The REXX exec in the following example uses CA Enterprise Tester session commands to start a session and to use transactions designed for users.

```
     /* Exec: OPEN ProTerm commands for REXX: */
     /* LOGON INVITE TYPE LOGOFF */
(1)      address ProTerm
(2)    "  logon cics 24x80 24x80 basicds "
       "  invite 1000 "
       "  type clear "
(3)    "  type 'cemt set dataset(somedd) open' enter "
(4)      if substr(PTEBUFF,224,6) = 'NORMAL' then
            say 'open worked ok'
         else
            say 'open failed!!!'
       "  type pf3 "
(5)    "  logoff "
```

Tracing Exec-Driven Sessions

CA Enterprise Tester TRACE and ENDTRACE commands can be used to capture an exec-driven CA Enterprise Tester session. When a session is traced, a screen image is captured in a trace data set whenever a TYPE command sends data to an online system. Another screen is captured when a response or other message sent by an online system is received at the virtual terminal used for a session.

CA Enterprise Tester commands for REXX, the Interactive System, the Script Language, and other components of CA Enterprise Tester use a single system to create, manage and capture sessions. A session trace captured using the TRACE command is identical to a session traced captured by any other component of CA Enterprise Tester. Like any other FORMAT trace, a trace produced using the TRACE command FORMAT option may be browsed, analyzed and processed using CA Enterprise Tester Interactive System Browse, Analyze and Utility panels.

In the following example, an exec named TRACE uses TRACE and ENDTRACE commands to capture a CICS session.

```
      /* Exec: TRACE ProTerm commands for REXX:       */
      /*       TRACE LOGON INVITE TYPE ENDTRACE       */
(1)     address TSO
       " allocate f(tracedd) sysout(a) "
(2)     address ProTerm
       " TRACE to tracedd image "
(3)    " LOGON cics1 model mod2 "
       " INVITE 1000 "
       " TYPE clear "
       " TYPE 'cemt set dataset(somedd) open' ",
       " enter "
         if substr(PTEBUFF,224,6) = 'NORMAL' then
            say 'open worked ok'
         else
            say 'open failed!!!'
       " TYPE pf3 "
       " TYPE home 'logoff' erase enter "
(4)    " ENDTRACE "
(5)     address TSO
       " free f(tracedd) "
```

CA Enterprise Tester Ports and Multiple Sessions

A CA Enterprise Tester port is a place to anchor a CA Enterprise Tester session, session trace, data queue and/or a REXX environment. Each CA Enterprise Tester port is identified by a number. When the first exec in an address space issues a command, the exec's REXX environment is anchored to CA Enterprise Tester port 1.

CA Enterprise Tester commands that operate on sessions, traces, execs or queues, create or use the session, trace, REXX environment or queue anchored to the numbered port being accessed by the exec when the command is issued. By default, an exec accesses the port to which it's REXX environment is anchored. If an exec issues a CA Enterprise Tester LOGON command, causing the exec's REXX environment to be anchored to port 1, the LOGON command allocates and anchors a virtual terminal to port 1 and uses the terminal to start a session on port 1.

While only one session may be active on a particular port at any point in time, an exec may use the ACCESS command to access another port and may use the LOGON command to start a session on the accessed port.

For example, ACCESS 2 instructs subsequent commands to create or use port 2 resources. While accessing port 2, a LOGON command would allocate a virtual terminal and use it to start a session on port 2, and TYPE commands would use the port 2 virtual terminal keyboard to enter transactions. An exec may use the ACCESS command multiple times to start multiple sessions on different ports, and may access a specific port at any time to enter transactions using the session anchored to the accessed port.

In addition to instructing subsequent commands to create or use resources anchored to a particular port, each ACCESS command also instructs session commands to return session information to an exec using either simple or compound session variable names.

- Simple variable names used by CA Enterprise Tester to return session information, including PTEBUFF, are defined in CA Enterprise Tester REXX Variables.
- Compound session variable names like PTEBUFF.1, or PTECOLS.3, have a REXX stem that is a defined simple session variable name and a tail that is the accessed port number.

By default, or if an exec issues an ACCESS EXECPORT command to access the port it's REXX environment is anchored to, session commands return session information to the exec using the defined simple session variable names.

If an exec issues an ACCESS NEWPORT command to access a previously unused port, or an ACCESS FREEPORT command to reuse a port that is no longer being used, or if an exec issues an ACCESS command that specifies a port number, including the number of the port that the exec's environment is anchored to, the ACCESS command instructs subsequent session commands to return session information to the exec using compound session variable names.

As a result, when an exec uses the ACCESS command to start and use multiple sessions on multiple ports, the exec may refer to returned session information using either simple variable names or using compound variable names that have an actual or variable port number as a tail, that refers to information about a particular session.

The REXX exec in the following example uses ACCESS commands to start three sessions on three ports, to run three long running end-of-day transactions concurrently. The exec also uses CA Enterprise Tester HANDLE and WAIT commands to control session flow and timing.

```
     /* Exec: MULTISES ProTerm commands for REXX: */
     /* ACCESS LOGON INVITE HANDLE WAIT */
        address ProTerm
(1)     sys.1 = 'cicsa'
        sys.2 = 'cicsb'
        sys.3 = 'cicsc'
        do port=1 to 3
      " ACCESS &port "
      " LOGON &sys.port model mod2 "
      " INVITE 1000 "
      " TYPE clear "
      " HANDLE turnaround off "
      " TYPE 'long running end-of-day tran' ",
      " enter "
      " HANDLE turnaround on "
        end
(2)   " WAIT 10000 on response "
        do until done = 3
           done = 0
           do port=1 to 3
              if PTEKB.port = 'INHIBITED' then do
      " ACCESS &port "
      " INVITE "
        end
      else
        done = done + 1
      end
    end
```

Sharing Sessions with Called Execs

When an exec calls another exec, the called exec runs in the same REXX environment as the calling exec, and by default accesses the port to which the REXX environment is anchored.

An exec may issue a LOGON command to start a session, then call another exec that issues a TYPE command to sign on. However, REXX variable pools are not shared by called and calling execs. When the called exec issues the TYPE command, PTEBUFF and other session variables are returned to the called exec, and PTEBUFF and other session information in the calling exec's variable pool becomes obsolete.

If an exec starts a session then calls another exec that uses the session, when the called exec returns, the calling exec may refresh session variables in it's REXX variable pool using the QUERY SESSION command.

In the following example LOGON issues a LOGON command, then calls SIGNON to sign on. When SIGNON returns, LOGON uses the QUERY SESSION command to determine the status of the session and to refresh PTEBUFF and other session information.

```
/* Exec: LOGON ProTerm commands for REXX: */
/* LOGON QUERY TYPE */
   address ProTerm
 " LOGON cics1 model mod2 "
   call 'SIGNON'
 " QUERY session "
   select
      when PTEINFO = 'NOSESSION' then do
         say 'session failed'
         exit
         end
      when substr(PTEBUFF,2,15),
         \= 'Signon Complete' then do
         say 'sign on failed'
         exit
         end
      otherwise
 " TYPE clear "
      end
/* etc. */
/* Exec: SIGNON ProTerm commands for REXX: */
/* INVITE TYPE */
   address ProTerm
 " INVITE 1000 "
 " TYPE 'uid1' token 'uid1a' enter "
    return
```

CA Enterprise Tester's resources are cleaned up automatically when the first exec that runs in the port 1 REXX environment ends. If an exec called another exec to start a session before it issued a CA Enterprise Tester command, the REXX environment would be anchored to port 1 when the called exec issued the LOGON command, and would be the first exec in the port 1 REXX environment. The session would be cleaned up automatically by CA Enterprise Tester when the called exec returned to the calling exec.

If no active exec in an address space is using CA Enterprise Tester commands, and an exec is executed that intends to use resources created by a called exec, it must issue some CA Enterprise Tester command, for example, ACCESS EXECPORT, before calling an exec that creates the resources, to ensure that the resources are not cleaned up when the called exec ends.

Attaching Asynchronous Exec Tasks

When an exec creates multiple sessions on multiple ports, then uses the ACCESS command to switch from session to session, the events on the multiple sessions occur serially or synchronously. That is, only one TYPE command can be active at a time. Each TYPE command must complete before the exec can ACCESS another port or execute another TYPE command. The same is true if an exec calls another exec, because the called exec must return before the calling exec can execute additional session commands.

The CA Enterprise Tester ATTACH command may be used to attach an exec that runs at the same time as the attaching exec. Unlike calling and called execs that run synchronously and in the same REXX Language Processor Environment, attaching and attached execs run asynchronously in different REXX Language Processor Environments and as separate z/OS tasks.

An ATTACH command specifies the name of the exec to be attached and may specify up to 20 literal or variable argument values to be passed to the attached exec. An ATTACH command may also specify REXX libraries to be used to load execs, the destination of REXX SAY messages issued by the attached exec and/or the destination of error messages issued by REXX while the exec is executing.

The ATTACH command creates a new z/OS task and a REXX Language Processor Environment that are used to run the attached exec. The REXX environment is anchored to the port being accessed by the attaching exec when the command is issued. Before issuing an ATTACH command, an ACCESS command must be issued to access a CA Enterprise Tester port that is not already associated with a REXX environment.

After loading the attached exec in the new REXX environment, the ATTACH command completes and the attaching exec resumes execution at the same time that the attached exec begins execution.

Attached execs run as peer-to-peer tasks and are not arranged in a hierarchy. No notification is given to an attaching task when an attached task ends. An attaching exec may end before an attached exec ends. Any exec in an address space may use the ATTACH command to attach any other exec and any attached exec may use any CA Enterprise Tester command to create CA Enterprise Tester resources or to use resources created by any other exec running on any port in the address space.

Sessions, traces, data queues and other CA Enterprise Tester resources created by execs are cleaned up automatically by CA Enterprise Tester only when the first exec running in the port 1 REXX environment end. When the port 1 exec ends, cleanup is delayed until any other active execs that have issued CA Enterprise Tester commands have ended.

In the following example an exec named FAST, attaches another exec named CHANGE 10 times to change a user password in 10 remote systems. The result would be the same if FAST called CHANGE10 times instead of attaching it. However, because called execs run one at a time, and because attached execs run at the same time, the ATTACH solution runs 10 times faster.

```
     /* Exec: FAST ProTerm commands for REXX: */
     /* ACCESS ATTACH */
(1)     arg uid oldpt newpt
        address ProTerm
        do n=1 to 10
(2)    " ACCESS newport "
       " ATTACH exec change ",
       " &uid &oldpt &newpt 'tso"n"' "
           end
     /* Exec: CHANGE ProTerm commands for REXX: */
     /* MONITOR LOGON INVITE TYPE */
(3)     arg uid oldpt newpt tson
        address ProTerm
       " MONITOR signals "
(4)    " LOGON &tson data &uid "
        do until substr(PTEBUFF,34,11),
           = 'TSO/E LOGON'
       " INVITE 1000 "
           end
(5)    " TYPE token &oldpt tab token &newpt enter "
        if substr(PTEBUFF,11,16)
           ª= 'REENTER PASSWORD' then do
       " TYPE token token &newpt enter "
          if substr(PTEBUFF,11,16)
          ª= 'PASSWORD CHANGED' then do
              say tson 'password changed'
          exit
             end
          end
       say tson 'password change failed'
```

Queues and Intertask Communication

Attached execs that run as asynchronous z/OS tasks may use CA Enterprise Tester data queues to communicate. Like the commands that attach execs and manage sessions, CA Enterprise Tester commands that manage data queues operate on the data queue anchored to the port being accessed by an exec when a command is executed.

QUEUE, PUSH, and POST Commands

QUEUE, PUSH and POST commands may store up to 20 literal or variable argument values as a single entry in an accessed queue. PULL and QUERY QUEUE commands return the argument values fetched from an accessed queue entry in corresponding REXX variables specified by the PULL or QUERY QUEUE command.

- The QUEUE command adds an entry to the bottom of a queue.
- The PUSH command adds an entry to the top of a queue.
- The POST command replaces an entire queue with a specified queue entry.
- The PULL command returns data values fetched from the top entry in a queue and deletes the entry from the queue.
- The QUERY QUEUE command returns data values fetched from the top entry in a queue, but leaves the entry in the queue.
- The DELETEQ command deletes an entire data queue.

Data queues can be used in any desired way:

- A queue may be used by a single exec as private temporary storage.
- A queue may be used as a bulletin board to post information that may be examined by other execs.
- A queue may be used to collect data that is stored by one or more execs and is processed by a single exec.
- A queue may be used to distribute data to multiple execs that access a common queue.
- Multiple queues may be used to distribute data to execs that access designated queues.

By default, if an accessed queue is empty when a PULL command is issued, RC=12is returned to the exec and the command completes immediately. An exec designed to process data queued by other execs may use the WAIT command to instruct subsequent PULL commands to wait a specified amount of time or to wait FOREVER for an entry to be queued, if the accessed queue is empty when a PULL command is issued.

The SERVER and SESSION execs in the following examples illustrate two uses of CA Enterprise Tester data queues. The SERVER exec attaches SESSION execs to multiple ports and distributes a file of data to the queues anchored to SESSION exec ports. Each SESSION exec starts a session and uses the session to process queue entries pulled from the queue anchored to the SESSION exec's port. After processing each distributed queue entry, the SESSION exec returns a status message to the queue anchored to the SERVER exec's port. After logging a status message pulled from the SERVER exec's queue, the SERVER exec distributes another record to the SESSION exec that returned the logged status message.

Attached execs that drive asynchronous sessions and communicate using data queues may be used to perform system stress testing or may be used in production applications to multiply the throughput possible using conventional serial data processing methods.

```
     /* Exec: SERVER ProTerm commands for REXX: */
     /* ACCESS QUEUE ATTACH WAIT PULL */
(1)     address TSO
       " alloc f(quefile) da('appl.quedata') shr "
       " alloc f(logfile) da('appl.logdata') mod "
         querecs = 0
(2)     do 10
address TSO
     " execio 1 diskr quefile      "
if rc \= 0 then do
querec = 'eof'
leave
end
pull querec
querecs = querecs + 1
address ProTerm
     " ACCESS newport      "
     " QUEUE &querec      "
     " ATTACH exec session &pteport      "
end
address ProTerm
     " WAIT forever on pull      "
(3) do while querec \= 'eof'
     " ACCESS execport      "
     " PULL &logrec &port      "
address TSO
queue logrec
     " execio 1 diskw logfile      "
querecs = querecs - 1
     " execio 1 diskr quefile      "
if rc = 0 then do
pull querec
querecs = querecs + 1
end
else
querec = 'eof'
address ProTerm
     " ACCESS &port      "
     " QUEUE &querec       "
end
(4) do while querecs > 0
     " ACCESS EXECPORT      "
     " PULL &logrec &port      "
address TSO
queue logrec
     " execio 1 diskw logfile      "
querecs = querecs - 1
address ProTerm
          " ACCESS &port           "
          " QUEUE 'eof'           "
end
(5) address TSO
          " execio 0 diskr quefile (FINIS           "
          " free f(quefile)           "
          " execio 0 diskw logfile (FINIS           "
          " free f(logfile)           "
```

SERVER exec notes:

1. The SERVER exec allocates a quefile that contains data to be processed, allocates a logfile used to collect status messages and initializes a count of in-progress quefile records to zero.

2. The SERVER exec startup loop uses EXECIO and the REXX PULL instruction to read records sequentially from the quefile, and counts each record read as an in-progress quefile record. For each record read, the CA Enterprise Tester ACCESS command is used to access an unused port, the QUEUE command is used to store the quefile record in the data queue anchored to the accessed port, and an ATTACH command is used to attach the SESSION exec to the accessed port to process quefile records distributed to the SESSION exec by the SERVER exec. An ATTACH parameter provides the SERVER exec's port number to each attached SESSION exec.

   SESSION execs use the SERVER exec's port number to return status messages to the SERVER exec after processing each distributed quefile record. Before processing status messages, the SERVER exec uses the CA Enterprise Tester WAIT command to instruct subsequent PULL commands to wait for a status message to be queued if the SERVER exec's queue is empty when a PULL command is issued.

3. Each iteration of the SERVER exec mainline loop logs a status message received from any attached SESSION exec and supplies another quefile record to the SESSION exec. The SERVER exec uses the ACCESS command to access the SERVER exec's port and uses the CA Enterprise Tester PULL command to wait for and pull an entry from the SERVER exec's queue. Each queue entry added to the SERVER exec's queue by an attached SESSION exec contains a status message and also the port number of the SESSION exec that added the entry to the SERVER exec's queue. The REXX QUEUE instruction and EXECIO are used to write each pulled status message to the logfile managed by the SERVER exec, and the count of in-progress quefile records is decremented after logging the status message for a processed quefile record.

   After logging a status message returned by a SESSION exec, the SERVER uses EXECIO and the REXX PULL instruction to read another quefile record to be processed by the SESSION exec, and counts each record read as an in-progress quefile record. When the end of the quefile is reached, an 'eof' record is created to inform the SESSION exec that all quefile records have been processed and to signal the end of the SERVER mainline loop. The SERVER exec uses the port number that was pulled with the previous status message to ACCESS the SESSION exec's port and uses the QUEUE command to store either the next quefile record or the eof record in the SESSION exec's queue.

4. When the end of the quefile has been reached, the SERVER exec pulls, logs and accounts for the status message for each in-progress quefile record, and queues an eof record to each SESSION exec that returns a final status message, until the count of in-progress quefile records is zero.
5. After all quefile records have been processed and all status messages have been logged, quefile and logfile are closed and freed, and the SERVER exec ends.

```
/* Exec: SESSION ProTerm commands for REXX: */
/* LOGON INVITE TYPE LOGOFF */
/* WAIT ACCESS PULL QUEUE */
(1) arg logport
address ProTerm
" LOGON 'cics1' mod2 "
" INVITE 1000 "
" TYPE 'uid' token 'uida' enter "
" TYPE clear "
" TYPE 'tran' enter "
" WAIT forever on pull "
(2) do forever
" ACCESS execport "
" PULL &querec "
if querec = 'eof' then
leave
" TYPE &querec enter "
logrec = substr(PTEBUFF,23*80+1,80)
" ACCESS &logport "
" QUEUE &logrec &pteport "
end
(3) " TYPE clear "
" LOGOFF
```

SESSION exec notes:

1. Each attached copy of the SESSION exec uses the REXX ARG instruction to get the SERVER exec port number passed to the SESSION exec as a parameter by the ATTACH command used by the SERVER exec to attach the SESSION exec.

   The SESSION exec starts a session, signs on, and enters a tran that could be any transaction that processes any data contained in any querec.

   The CA Enterprise Tester WAIT command is used to instruct subsequent PULL commands to wait for a querec to be queued if the SESSION exec's queue is empty when a PULL command is issued.
2. Each iteration of the SESSION exec mainline uses ACCESS and PULL commands to wait for and to pull a querec from the SESSION exec's queue, uses CA Enterprise Tester session commands to enter the pulled querec, and uses ACCESS and QUEUE commands to store an entry in the SERVER exec’s queue. The queue entry includes a status message displayed by 'tran' on screen row 24, and also includes the port number of the SESSION exec that stored the queue entry.
3. An eofquerec instructs the SESSION exec to clean up it's session and end.

Sharing Resources with Attached Execs

Attached execs run as z/OS tasks and may be executing at the same time using multiple hardware processors. When attached execs share CA Enterprise Tester resources and multiple execs attempt to use a shared resource at exactly the same time, processor access to the shared resource is serialized automatically. For example, if two execs attempt to pull data from the same queue at the same time, the requests are processed serially and each exec pulls a different entry from the queue.

Locks are used to serialize access to resources. When an exec issues a CA Enterprise Tester command that uses a shared resource, a lock is obtained that permits exclusive use of the resource by the exec task that owns the lock, before the resource is used. If an exec issues a command that uses a shared resource, while the resource lock is owned by another task, the task that wants the lock queues a lock requests and waits for the lock.

After a requested operation has been performed on a locked resource, if a lock request was queued by another exec task, ownership of the resource lock is transferred to the task that queued the oldest lock request and execution of the task that then owns the lock is resumed, otherwise the resource is unlocked.

Resource locks are obtained automatically by CA Enterprise Tester commands when shared resources are used, but are held only for the life of the command. In some applications it may be necessary or convenient to hold resource locks while issuing a series of commands. An exec may use the CA Enterprise Tester LOCK command or an ACCESS command that specifies the LOCK keyword to lock an accessed port. When a port is locked, the port and all resources anchored to the port are locked. A port lock is held until an UNLOCK command is issued.

If an exec issues commands that use resources anchored to a port that is locked by another exec task, or attempts to lock a port that is locked by another exec task, lock requests are queued and remain queued until the exec that holds the port lock unlocks the port.

An exec may lock a port to issue a series of commands that use resources anchored to the port, while preventing other execs from using port resources while the commands are executed. For example:

- An ACCESS FREEPORT LOCK command accesses a port that is devoid of resources and locks the port until an UNLOCK command is issued. Two or more execs may issue the command at the same time to attach other execs to the accessed ports without the possibility that the ACCESS command would assign the same FREEPORT to two execs before either had a chance to issue the ATTACH command.

- If an exec task is used to log records that are queued by multiple other exec tasks, and it is desirable to log a series of related records that are queued by a single exec task without interleaving the records with records queued by other exec tasks, LOCK and UNLOCK commands may be used by the execs that queue records, to prevent other exec tasks from queuing records while a series of related records are being queued.

Smart Exec-Driven Sessions

Recorded REXX execs created by capturing interactive user sessions contain CA Enterprise Tester LOGON, TYPE, INVITE and DELAY commands that start sessions, re-key user keystrokes, and that handle other detected session events. When they are executed, recorded REXX execs create sessions that are similar to the captured user session, provided that online system responses to re-keyed input are similar to online system responses to original input.

While recorded REXX execs repeat only what a person did, and do not know why they did it or what they would have done if a system had responded to input in a different way, when a recorded exec is executed, the virtual terminal screen images returned by session commands in REXX variables provides the same information to an exec that displayed online system responses provide to a terminal user.

REXX instructions and functions, returned session information and other CA Enterprise Tester commands and command options may be used to develop intelligent REXX execs that make the decisions that people do. For example:

- REXX execs may examine virtual terminal screen images to verify that transactions completed successfully and to decide what to do next.
- Information from previous online system responses may be saved and used as input to subsequent transactions.
- The CA Enterprise Tester QUERY command may be used to check the status of or request information about CA Enterprise Tester resources.
- The CA Enterprise Tester SIGNAL command may be used to trap session failures or other unusual events using REXX condition traps or return codes may be checked to detect the events.
- The CA Enterprise Tester WAIT command may be used to interrupt session commands if an online system response is not received in a reasonable amount of time.

Intelligent REXX applications that use CA Enterprise Tester session commands and any other resources available to REXX execs may be used to perform or automate almost any imaginable terminal task that could be performed by a person using a terminal.

ISPF Session Panels

Options of the CA Enterprise Tester LOGON and TYPE commands make it easy to display CA Enterprise Tester virtual terminal screen images using an ISPF panel and to rekey data entered at the displayed panel into the virtual terminal display.

The LOGON command PANELDATA keyword instructs subsequent CA Enterprise Tester session commands to edit the virtual terminal display buffer returned in variable, PTEBUFF, for display using an ISPF panel similar to the panel shown in the following example.

```
)ATTR
01 TYPE(DATAIN)  INTENS(LOW)  SKIP(OFF) CAPS(OFF) JUST(ASIS)
05 TYPE(DATAIN)  INTENS(HIGH) SKIP(OFF) CAPS(OFF) JUST(ASIS)
07 TYPE(DATAIN)  INTENS(NON)  SKIP(OFF) CAPS(OFF) JUST(ASIS)
11 TYPE(DATAOUT) INTENS(LOW)  SKIP(OFF) CAPS(OFF) JUST(ASIS)
15 TYPE(DATAOUT) INTENS(HIGH) SKIP(OFF) CAPS(OFF) JUST(ASIS)
19 TYPE(DATAOUT) INTENS(LOW)  SKIP(ON)  CAPS(OFF) JUST(ASIS)
1D TYPE(DATAOUT) INTENS(HIGH) SKIP(ON)  CAPS(OFF) JUST(ASIS)
| AREA(DYNAMIC)  EXTEND(ON)   USERMOD(3F)
)BODY WIDTH(80)
|PTEBUFF |
)INIT
.ALARM = &PTEALARM
)PROC
&CSRPOS = .CSRPOS
&PFKEY = .PFKEY
)END
```

1. When a LOGON command specifies PANELDATA, basic 3270 attributes in the virtual terminal display buffer are returned in PTEBUFF as one of the seven attributes in the session panel definition. 3270 Attribute Values documents the mapping of basic 3270 attributes to panel attributes.
2. PANELDATA also edits PTEBUFF data to remove nulls in protected fields that would otherwise be displayed as periods by ISPF.
3. When the session panel is used to display a virtual terminal screen image, ISPF displays the PTEBUFF variable value returned by CA Enterprise Tester session commands, in the PTEBUFF dynamic area defined in the body of the session panel.
4. The USERMOD(3F) specification in the panel definition instructs ISPF to replace a field attribute in the PTEBUFF variable with the hexadecimal value, 3F, when data is keyed into a field. In a CA Enterprise Tester LOGON command, USERMOD '3F' informs CA Enterprise Tester of the USERMOD value defined in a session panel.

5. CA Enterprise Tester session commands return YES in variable, PTEALARM, when a message received from an online system sounds the terminal alarm. The reference to PTEALARM in the panel definition causes ISPF to echo an alarm at the terminal if an alarm was sounded at the virtual terminal.
6. The PROC section of the panel definition instructs ISPF to return the cursor position and the PF key in variable fields CSRPOS and PFKEY, when data is entered and the ISPF DISPLAY command completes.

The exec named SESSMAN, shown in the following example is a TSO command that starts and manages an interactive session with a named online system. For example, 'TSO SESSMAN CICSA', may be entered while using any ISPF panel to start a session with an onlinesystem, CICSA. The SESSMAN exec uses a session panel, the ISPF DISPLAY command, and CA Enterprise Tester session commands to start and display a virtual terminal session at a real terminal and to rekey data entered at a terminal into the virtual terminal display and send the data to an online system. To the user, a session managed by SESSMAN works much like any other terminal session.

```
    /* Exec: SESSMAN ProTerm commands for REXX: *    /
    /* MONITOR SIGNAL LOGON DELAY TYPE *    /
(1) arg system
address ProTerm
(2)      " monitor signals      "
     " signal error on logoff      "
(3)      " logon &system paneldata usermod '3F'      ",
     " 24x80 24x80 basicds      "
(4)      " delay 500      "
     " signal off on logoff      "
do forever
address ispexec ,
(4)      " display panel(sesspan)      ",
     " cursor(ptebuff) csrpos(     "ptecsrp     ")      "
(5) if pfkey = ' ' then
aid = 'enter'
else
aid = pfkey
     " type usermod <&csrpos> aid &aid      "
(6) if rc     /= 0 then
leave
end
```

SESSMAN Exec Notes:

1. The name of an online system specified in a SESSMAN command is received by the SESSMAN exec as a REXX argument.
2. CA Enterprise Tester MONITOR and SIGNAL commands are used to display an error message if a requested session can not be started.
3. A CA Enterprise Tester LOGON command is used to start a session with the named online system. The command specifies PANELDATA to instruct subsequent session commands to edit PTEBUFF for display using a session panel, and specifies USERMOD '3F' to inform CA Enterprise Tester of the USERMOD value defined by the session panel.
4. After the initial sign on screen is received by the DELAY command, the SESSMAN exec enters a DO loop that ends only when the session ends.

5. An image of each display received at the virtual terminal is displayed at the TSO terminal using the ISPF DISPLAY command and a session panel like the panel shown in the following example. The DISPLAY command positions the cursor using the virtual terminal cursor position returned by session commands in variable, PTECSRP.
6. A TYPE command that specifies the USERMOD keyword is used to rekey data entered by the terminal user into the virtual terminal display. USERMOD finds data in PTEBUFF that was entered by the user using the USERMOD value specified by the LOGON command and in the session panel definition. The TYPE command uses the ENTER key or the PF key pressed by the terminal user to send the input to the online system.
7. The session and the SESSMAN exec end when the user enters a logoff transaction that causes the online system to end the session.

Monitoring Execs and Sessions

CA Enterprise Tester commands are designed to automate terminal tasks and provide options that allow REXX execs to handle all possible errors and other events, and therefore, by default, do not automatically display error messages, exec-driven sessions or other command events. However, an exec or an exec user may use command monitoring functions built into CA Enterprise Tester commands for REXX, to monitor selected events and to document, display or otherwise process errors, exec-driven sessions and other events automatically, while an exec is executing.

An interactive utility, the CA Enterprise Tester Command Monitor For REXX, may be used to execute and monitor a recorded or user-written REXX exec that uses CA Enterprise Tester session commands, to display error messages, commands, exec-driven session events and other information at a terminal, while the exec is executing. The Command Monitor may also be used to manage a system stress test or other multi-session online system test that is driven by a list of execs that create CA Enterprise Tester sessions. When a list of execs is specified, the Command Monitor attaches each exec as an asynchronous z/OS task, and displays virtual terminal screen images and other events driven by the attached execs in windows, allowing the user to view multiple concurrent session events at a single terminal at the same time, while the test is in progress. Refer to The CA Enterprise Tester Command Monitor for REXX for complete information about Command Monitor services.

The Command Monitor uses the CA Enterprise Tester MONITOR command to activate the general purpose command monitoring functions. The MONITOR command may be used in any exec to automatically display information about specified events while an exec is executing or to call a monitor exec to process the information:

- MONITOR SIGNALS detects REXX conditions raises by CA Enterprise Tester commands and may be added to any exec to display messages automatically when REXX conditions are raised.
- MONITOR CONDITIONS detects non-zero return codes returned by CA Enterprise Tester commands and may be used to display messages that document the reason.
- MONITOR COMMANDS may be used to display CA Enterprise Tester commands as they are issued by REXX execs.
- MONITOR RESPONSES calls a default or specified monitor exec when responses or other outbound session events are received and may be used to display an exec-driven session at a terminal.
- MONITOR STORE, DROP, FETCH and/or one or more generic variable names, specified in any combination, may be used to display variable values when they are changed, used or dropped by CA Enterprise Tester commands.

MONITOR command VIASAY, VIATPUT and VIAEXEC options specify how monitored events are handled when they are detected. VIASAY and VIATPUT display events other than RESPONSES using the REXX SAY instruction or TSO TPUT, depending on the REXX environment in which the exec is executing. VIAEXEC calls a specified monitor exec when a monitored event is detected. When RESPONSES are monitored, a monitor exec is always called to process online system responses received by CA Enterprise Tester session commands, either an exec specified by the VIAEXEC option or a default monitor exec.

When a monitor exec is called to process detected events, call arguments provide detailed information about the event to the monitor exec. Call arguments are explained in The Monitor Exec Interface. The monitor exec shown in The Distributed Monitor Exec PTEMONX is designed to be used as the default monitor exec that is called when RESPONSES are monitored but a VIAEXEC is not specified. When it is called to process an online system response to a transaction entered by an exec using the CA Enterprise Tester TYPE command, PTEMONX uses either ISPF display services or the REXX SAY instruction to display a virtual terminal screen image, depending on the REXX environment in which the exec is executing.

A MONITOR command applies to all execs that execute in the REXX environment in which the MONITOR command is executed, and may be issued by a general purpose front-end exec that is used to call and monitor other execs with no modification to the called execs. The CA Enterprise Tester Command Monitor for REXX is an example of a front-end exec that uses the MONITOR command to monitor called execs. However, the Command Monitor is an ISPF Dialog and can only be used to execute and monitor execs in the ISPF environment.

Another front-end exec, PTESMON, is shown in the following example. PTESMON may be used to call and monitor an exec in REXX environments created by ISPF, TSO or IRXJCL, with no modification to the called exec, by entering the following as a TSO command or as an IRXJCL PARM:

```
PTESMON testexec args
```

PTESMON issues a MONITOR command to monitor CA Enterprise Tester commands issued by the called exec, testexec, relying on the default monitor exec, PTEMONX, to display response events. Both PTESMON and PTEMONX use either ISPF display services or the REXX SAY instruction to display events depending on the REXX environment. When a specified testexec returns, PTESMON reports any REXX condition caused by the call to testexec and ends any session not ended by testexec.

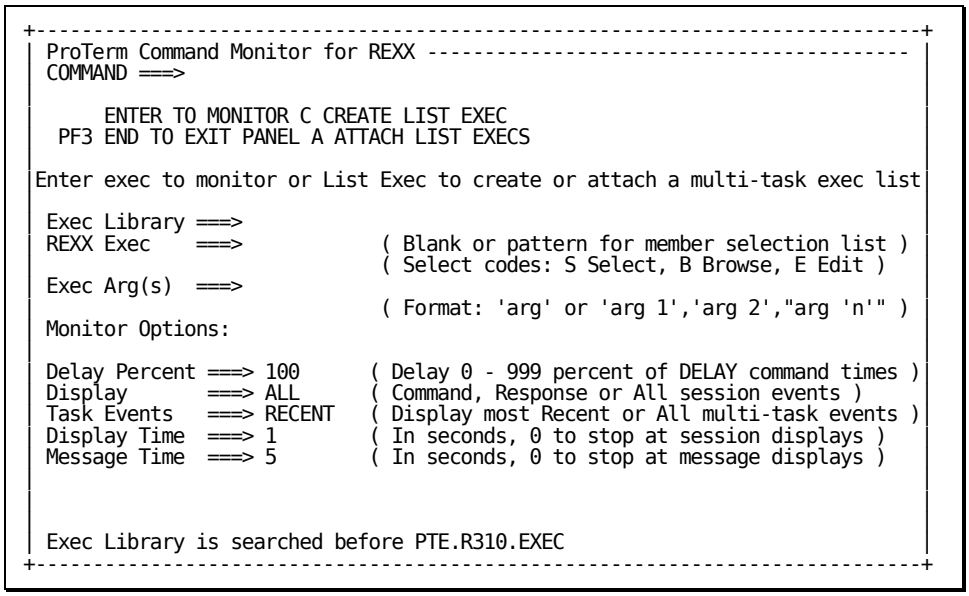
The following is an example of the PTESMON exec-driven session monitor:

```
    /* PTESMON - A Monitor for REXX-driven ProTerm Sessions (distributed) */
        arg args
        parse upper var args execname execargs
        if execname = '' then do
            say 'Enter: PTESMON execname (optional args for execname)'
            exit
            end
        address ProTerm
      " ACCESS EXECPORT " ; " QUERY SESSION "
        if ptestate \= 'PTESTATE' then " LOGOFF "
      " MONITOR SIGNALS RESPONSES "
          signal on halt name halt     ; signal on error name error
          signal on syntax name syntax ; signal on failure name fail
          interpret "call '"execname"' '"execargs"'"
          problem = ''                 ; signal finish
     halt: problem = 'HALT'            ; signal finish
    error: problem = 'ERROR'           ; signal finish
   syntax: problem = 'SYNTAX'          ; signal finish
     fail: problem = 'FAILURE'         ; signal finish
   finish: signal off halt             ; signal off error
          signal off syntax            ; signal off failure
     " MONITOR OFF " ; " ACCESS LASTPORT " ; " QUERY SESSION "
     parse source . . . . . . addrspn .
     if addrspn = 'ISPF' then do
          green = '01'x ; red = '05'x ; blue = '11'x ; white = '15'x
          PTEDESKT = ''
          address ispexec
          " control nondispl " ; " display panel(PTEPXMDT) "
          zwinttl = execname 'execution ended on Port' PTEPORT
          select
               when problem \= '' then
                  PTEMWIN2 = left(red || execname || white,
                                  || 'terminated; REXX condition(',
                                  || red || problem || white || ').',72),
                                  || right(blue || 'Press' || green,
                                  || 'ENTER' || blue || 'to end test.',72)
          when ptestate.pteaport \= 'PTESTATE.'PTEAPORT then
                    PTEMWIN2 = left(red || execname,
                                  || white || 'did not end Port' PTEAPORT,
        'session. ' red'LOGOFF issued.',72),
                                  || right(blue || 'Press' || green,
                                  || 'ENTER' || blue || 'to end test.',72)
          otherwise do
                                PTEMWIN2 = left(green || execname,
                                  || white || 'ended normally.',72),
                                  || right(blue || 'Test ending . . .',72)
          " control display lock "
              end
             end
          " addpop row(19) column(1) " ; " display panel(PTEPXMW2) "
          address ProTerm " WAIT 800 "
              end
        else do
           say copies('_',79) ; say ' '
           select
               when problem \= '' then
                     say execname 'terminated; REXX condition(' problem ').'
               when ptestate.pteaport \= 'PTESTATE.'PTEAPORT then
                     say execname 'ended but did not end Port',
                              PTEAPORT 'session. LOGOFF issued.'
                     otherwise say execname 'ended normally.'
                     end
                   end
               if ptestate.pteaport \= 'PTESTATE.'pteaport then " LOGOFF "
```

The CA Enterprise Tester Command Monitor for REXX

The CA Enterprise Tester Command Monitor for REXX is an interactive utility that may be used to execute and monitor a recorded or user-written REXX exec that uses CA Enterprise Tester commands, or to attach and monitor a list of execs that use CA Enterprise Tester commands, and to display commands, exec-driven sessions and other events as they occur while the exec or execs are executing. When a list of execs is attached and monitored, windows are used to display multiple concurrent exec-driven events at a single terminal at the same time.

To use the Command Monitor for REXX, enter the UTILITY command at the CA Enterprise Tester Control Panel and select Utility Menu option M to display the CA Enterprise Tester Command Monitor for REXX request panel.



Command Monitor Request Panel Services

To display an exec library member selection list:

1. Enter an Exec Library name;
2. Leave the REXX Exec field blank or enter a member name pattern; and
3. Press ENTER. While using a member selection list, members may be selected, browsed, or edited using S, B or E member select codes.

To execute and monitor a REXX exec:

1. Enter the Exec Library name;
2. Enter the name of the REXX Exec to be executed and monitored or select the exec from a member list;
3. Enter any Exec Arg(s) to be passed to the exec;
4. Enter Monitor Options (documented below); and
5. Press ENTER.

When a single REXX exec is monitored a full-screen image of the virtual terminal screen used for an exec-driven session is displayed when session commands are issued by the exec and/or when online system responses are received by the exec, as specified by Monitor Options.

To create a model List Exec:

1. Enter an Exec Library name;
2. Enter the name of the List Exec to be created as the REXX Exec; and
3. Enter the Create command.

A List Exec is an exec that returns a list of exec names to be attached and monitored using the Multi-Task Command Monitor. A specified List Exec is called when the ATTACH command is entered at the Command Monitor request panel. The list returned by a List Exec is single string of data that contains exec names separated by at least one blank, and may contain one or more arguments following each exec name to pass the arguments to the exec when it is executed. Argument values are enclosed in either single or double quotes. Multiple arguments for a single exec are separated by commas if commas would be coded in a REXX CALL instruction to pass the arguments to the exec.

The CREATE command stores a copy of the model List Exec shown below, as a member of the specified Exec Library using the specified REXX Exec name as the member name, and displays the member using ISPF EDIT. The model List Exec returns a list that contains 3 exec names, EXEC1, EXEC2 and EXEC3, and CALL arguments for EXEC2and EXEC3. A model List Exec is just a sample and must be edited to return the list of execs that are to be attached and monitored when the List Exec and the ATTACH command are entered at the Command Monitor panel.

```
exec_list = " EXEC1  "
 , " EXEC2 'arg 1' " , " EXEC3 'arg 1', 'arg 2' "
return exec_list
```

To attach and monitor a list of execs:

1. Enter the List Exec Exec Library name;
2. Enter the name of the List Exec that returns the list of execs to be executed as the REXX Exec or select the List Exec from a member list;

3. Enter any Exec Arg(s) to be passed to the List Exec;
4. Enter Monitor Options (documented below); and
5. Enter the ATTACH command.

The ATTACH command calls the specified List Exec and executes each exec in the list returned by the List Exec in a new REXX Language Processor Environment that runs as a new z/OS task. The number of REXX execs that may be attached is limited by the REXX Language Processor Environment limit contained in the TSO/E REXX environment table, IRXANCHR. IBM publication, TSO/E REXX/z/OS Reference, explains how to change the REXX environment limit.

Attached execs are monitored and commands and system responses specified by Monitor Options are displayed in windows to display multiple concurrent events at the same time. Task and session statistics are displayed while execs are active and summarized when all execs end.

Command Monitor Options

Delay Percent controls the speed of execs that issue DELAY commands by delaying a percentage of the time specified by each DELAY command. When less than 100% is specified, exec delays are shorter and execs run faster. When more than 100% is specified, delays are longer and execs run slower.

The Display option specifies the events to be displayed by the monitor. Command displays each session command issued by the exec and an image of the virtual terminal screen just before the command is executed. Response displays an image of the virtual terminal screen when an online system response or other outbound message is received at a virtual terminal, and does not display commands. All displays both session commands and system responses.

The Task Events option instructs the Command Monitor to display All or only the most Recent commands issued by exec tasks and/or online system responses. Attached execs run as asynchronous z/OS tasks and are not synchronized with other attached execs or with the Command Monitor. Attached execs continue to execute when exec-driven events are displayed by the monitor. When All task events are displayed, events are queued and displayed in the order that they occur. If a list of attached execs drive events faster than they can be displayed, the events displayed may not reflect current activity, and storage problems may occur if the event queue becomes very large. When Recent task events are displayed, the monitor displays only the most recent exec task commands and responses and discards older events when events occur faster than they can be displayed.

Display Time is the number of seconds the monitor waits after displaying a session event before selecting and displaying another event. If 0 (zero) is specified, the monitor waits for the user to press ENTER after displaying each session event.

Message Time is the number of seconds the monitor waits after displaying a message about an unusual event or error, before selecting and displaying another event. If 0 (zero) is specified, the monitor waits for the user to press ENTER after displaying any message.

Chapter 11: REXX Reference

This chapter documents the function and syntax of each host command used in the CA Enterprise Tester Host Command Environment.

- CA Enterprise Tester REXX Reference
- Syntax Notation Format
- ACCESS
- ATTACH
- CANCEL
- DELAY
- DELETEQ
- ENDTRACE
- HANDLE
- INVITE
- LOCK
- LOGOFF
- LOGON
- MONITOR
- POST
- PULL
- PUSH
- QUERY
- QUEUE
- SIGNAL
- TRACE
- TYPE
- UNLOCK
- WAIT

CA Enterprise Tester REXX Reference

The CA Enterprise Tester REXX Reference contains comprehensive documentation of all CA Enterprise Tester commands for REXX. The commands are presented in alphabetical order according to command name.

Syntax Notation Format

- CA Enterprise Tester Host Commands and keywords are in UPPERCASE text.
- User data is listed in lowercase bold text and can be literal data or variables.
- Optional parameters are enclosed in parentheses.
- Alternative keywords and/or parameters are separated by the bar symbol.
- In CA Enterprise Tester commands, REXX variable names are coded with a leading ampersand, &, to distinguish them from keywords and other command operands.
- Host command continuation is indicated with a comma.

ACCESS

When CA Enterprise Tester commands start or use sessions, start or end session traces, queue or pull queued data or perform other command functions, by default, they anchor or use resources anchored to the CA Enterprise Tester port on which the issuing exec is executing, and session commands return session information using the simple session variable names defined in CA Enterprise Tester REXX Variables.

The command

- Instructs subsequent CA Enterprise Tester commands to anchor or use resources anchored to a specified port when performing a command function.
- Instructs subsequent session commands to return session information using either simple or compound session variable names.
- Returns the accessed port number in REXX variable, PTEAPORT.
- Returns the current exec port number in PTEPORT.

Compound session variable names, like PTEBUFF.1, have a REXX stem that is a defined simple session variable name and a tail that is the accessed port number.

Examples

```
" ACCESS EXECPORT "
" ACCESS NEWPORT  "
" ACCESS 3 "
" ACCESS &port "
```

Format

```
ACCESS ( EXECPORT | * | port |  NEWPORT | FREEPORT | LASTPORT )  ( LOCK )
```

ACCESS EXECPORT or ACCESS *

Instructs subsequent commands to anchor or use resources anchored to the port on which the exec is executing, and instructs session commands to return information using simple session variable names.

ACCESS port

Where port is either a port number or a REXX variable name, coded with a leading ampersand, that contains a port number, instructs subsequent commands to anchor or use resources anchored to the specified port, and instructs session commands to return information using compound session variable names.

ACCESS NEWPORT(the default)

Assigns an unused port number, instructs subsequent commands to anchor or use resources anchored to the assigned port, and instructs session commands to return information using compound session variable names.

ACCESS FREEPORT

Locates a port on which no exec, session, session trace, data queue or lock is active, or assigns an unused port number, instructs subsequent commands to anchor or use resources anchored to the located or assigned port, and instructs session commands to return information using compound session variable names.

ACCESS LASTPORT

Identifies the port that was last accessed by any exec running on the current exec port, instructs subsequent commands to anchor or use resources anchored to the identified port, and instructs session commands to return information using compound session variable names.

LOCK

Can be specified to lock the accessed port. When a port is locked, an UNLOCK command must be issued before port resources can be created or used by an exec running on any port other than the port that holds the lock.

Conditions and Return Codes

At the completion of an ACCESS command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. The following table describes ACCESS command return codes and default REXX conditions that may be raised for some conditions.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
CANCEL	24	HALT	Cancel command processed
FAULT	64	HALT	Invalid command or system error

ATTACH

The ATTACH command creates a new z/OS task and REXX Language Processor environment, anchors the environment to the CA Enterprise Tester port being accessed by the exec when the ATTACH command is issued, and executes a specified exec in the new REXX Language Processor Environment.

The difference between calling an exec and attaching an exec is that the REXX CALL instruction completes and the calling exec resumes execution only when the called exec returns to the calling exec. However, the ATTACH command completes and the attaching exec resumes execution at the same time that the attached exec begins execution. Called and calling execs run one at a time, while attached and attaching execs run at the same time.

Examples

```
" ATTACH EXEC &execname    "," &arg1 'arg2' "

" ATTACH EXEC subexec 'logon' "" &sysid &uid &code  "
```

Format

```
ATTACH EXEC execname ( arg1 . . . arg20 )  ( EXECDD execdd )  ( SAYDD saydd )  (
SAYDEST saydest )  ( MSGDEST msgdest )
```

EXEC

A required keyword and it indicates that the value that follows is the name of the exec to be attached.

execname

The name of the REXX exec to be attached, and may be the actual exec name, a literal whose value is the exec name, or a variable name coded with a leading ampersand, for example, &execname, that contains the exec name.

arg1 through arg20

Up to 20 arguments to be passed to the attached exec. arguments to be passed to the attached exec. Each argument may be a literal value enclosed in single or double quotes, or may be a variable name, coded with a leading ampersand, for example, &arg, that contains an argument value.

Note: An attached exec may use a REXX ARG instruction to assign argument values to corresponding variables specified by REXX ARG instruction. ARG instruction variable names are separated by commas.

EXECDD

An optional keyword and indicates that the value that follows is the the DD name of the the REXX library(s) from which the attached exec and other execs are to be loaded in the REXX Language Processor Environment created by the ATTACH command.

execdd

The exec library DD name from which execs are loaded, and may be the actual DD name, a literal whose value is the DD name, or a variable name coded with a leading ampersand, for example, &execdd, that contains the exec library DD name. If an EXECDD is not specified, execs are loaded from the exec libraries used in the first REXX Language Processor Environment in an address space that uses CA Enterprise Tester commands for REXX, usually from SYSEXEC and/or SYSPROC.

SAYDD

An optional keyword and indicates that the value that follows is the DD name, saydd, to which REXX SAY messages are to be written in the REXX Language Processor Environment created by the ATTACH command. The SAYDD keyword option is subject to system options, described below, that control REXX message management options.

Saydd

The DD name to which REXX SAY messages are to be written and may be the actual DD name, a literal whose value is the DD name, or a variable name coded with a leading ampersand, for example, &saydd, that contains the DD name.

A null-valued variable may be specified as a sayddto request allocation of a new message dataset using a unique system-assigned DD name, and that the assigned name is to be returned to the attaching exec in the specified saydd variable.

If a specified saydd is already allocated when an ATTACH command is executed, the allocated dataset is used in the new REXX Language Processor Environment as a message dataset. If a saydd that was not allocated by CA Enterprise Tester is specified for two or more attached execs that run concurrently, the user must insure that the allocated dataset can be used as a shared message destination by the REXX Language Processor Environments created to run the attached execs.

SAYDEST

An optional keyword and indicates that the value that follows, saydest, is to be allocated as the destination of REXX SAY messages, if the specified or default SAYDD is not allocated when the ATTACH command is executed. The SAYDEST keyword option is subject to system options, described below, that control REXX message management options.

Saydest

The SAYDEST destination value and may be a JES SYSOUT class, an asterisk, *, to specify the default SYSOUT class for the address space, or may be the keyword, TERMINAL to send REXX SAY messages to a TSO terminal. Say dest may be the actual JES SYSOUT class or the TERMINAL keyword, may be a literal whose value is the SAY message destination, or may be variable name coded with a leading ampersand, for example, &saydd, that contains the SYSOUT class or TERMINAL keyword.

MSGDEST

An optional keyword and indicates that the value that follows, msgdest, is the destination of REXX error messages issued while processing execs in the REXX Language Processor Environment created by the ATTACH command. The MSGDEST keyword option is subject to system options, described below, that control REXX message management options.

msgdest

A REXX error message destination identified by the MSGDEST keyword, and is one of the destination keywords:

- WTO to write REXX messages using WTO only,
- SAY to write REXX messages to the REXX SAYmessage destination only; ATTACH
- ALL to write REXX messages using both WTO and to the REXX SAY message destination;
- OFF to discard REXX error messages.

Msgdest may be a destination keyword, may be a literal whose value is a destination keyword, or may be a variable name coded with a leading ampersand, for example, &msgdest, that contains a REXX message destination keyword.

Note: The number of REXX execs that may be attached is limited by the REXX Language Processor Environment limit contained in the TSO/E REXX environment table, IRXANCHR. IBM publication, TSO/E REXX/z/OS Reference, explains how to change the REXX environment limit.

All REXX Language Processor Environments in which CA Enterprise Tester commands are used are anchored to a CA Enterprise Tester port. When commands are used by an exec running in a REXX environment not created by CA Enterprise Tester, the environment is anchored to a port when the first command is issued. Only one REXX Language Processor Environment may be anchored to a port. The ATTACH command creates a new REXX Language Processor Environment and anchors the environment to the port being accessed by an exec anchored to the port. Before using the ATTACH command, an exec must use the CA Enterprise Tester ACCESS command to access a NEWPORT, a FREEPORT, or to access a specific port that is not already associated with a REXX Language Processor Environment.

An attached exec may ADDRESS PROTERM Tester and issue commands for REXX.

REXX Language Processor Environments created by the ATTACH command are not integrated into TSO. Attached execs may not use TSO and ISPF commands that are supported only in REXX Language Processor Environments that are integrated into TSO. However, execs that operate correctly when attached in a TSO address space may also be attached and operate correctly when IRXJCL is used to execute an exec as a batch job or when they are attached in other non-TSO address spaces.

Attaching Asynchronous Exec Tasks on page 10-8 and Queues and Intertask Communication on page 10-11 describe applications of the ATTACH command and other CA Enterprise Tester commands designed for use by execs that run in a multi-task environment.

REXX Messages Management Options:

The operation of and default values of the SAYDD, SAYDEST and MSGDEST keyword options are subject to CA Enterprise Tester system options that allow an installation to specify REXX message management options for each named address space in which CA Enterprise Tester commands for REXX are used. The name of an address space is defined by the REXX parameters module used to initialize REXX Language Processor Environments.

- System options may instruct CA Enterprise Tester to ignore ATTACH command SAYDEST and SAYDD specifications in address spaces that manage REXX message destinations for all REXX Language Processor Environments in the address space.

- System options may instruct CA Enterprise Tester to manage REXX message destinations only when an ATTACH command specifies SAYDEST and/or SAYDD options.
- System options may instruct CA Enterprise Tester to ignore any MSGDEST specified by an ATTACH command, and to use a specified option when running in a named address space.
- If system options permit, CA Enterprise Tester allocates and manages REXX message destinations and options as specified by ATTACH command SAYDEST, SAYDD and/or MSGDEST options or allocates and manages REXX message destinations using default values when ATTACH command options are not specified. Message datasets allocated by CA Enterprise Tester are automatically freed when the last attached exec using a message dataset ends.

Conditions and Return Codes

At the completion of an ATTACH command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. Table 11-2describes ATTACH command return codes and default REXX conditions that may be raised for some CA Enterprise Tester conditions.

Conditions	RC	Default Signal	Reason for Condition
LOCKOUT	4		Timed out waiting for port lock
CANCEL	24	HALT	CANCEL command processed
FAULT	64	HALT	Invalid command or system error

CANCEL

The CANCEL command causes the REXX HALT condition to be raised for the exec that is active on the port being accessed by the issuing exec.

Example

```
" CANCEL      "
```

Format

```
CANCEL
```

When the CANCEL command is executed, if an exec is active on the port being accessed, then cancellation of the exec is scheduled. The REXX HALT condition is subsequently raised when the cancelled exec recognizes the request following the execution of any CA Enterprise Tester command.

When the CANCEL command is executed, if a CA Enterprise Tester command is currently executing within the cancelled exec, and if the command is waiting for the completion of an event that is under CA Enterprise Tester control, then the wait for the completion of the event is interrupted immediately. The REXX HALT condition is raised when the interrupted command ends.

If an exec is not active on the port being accessed when a CANCEL command is executed, then no action is performed, but the command is considered to have completed without error.

Conditions and Return Codes

At the completion of an CANCEL command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. Table 11-3 describes CANCEL command return codes and default REXX conditions that may be raised for some CA Enterprise Tester conditions.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
CANCEL	24	HALT	CANCEL command processed
FAULT	64	HALT	Invalid command or system error

DELAY

The DELAY command delays execution of an exec for a specified amount of time, but allows outbound session traffic to flow on the session anchored to the port being accessed by the exec, during the delay.

Examples

" DELAY 3200 "" DELAY PERCENT 50 "

Format

DELAY hhmmssth | (PERCENT percent)

Hhmmssth

A delay time interval, expressed in hours (hh), minutes (mm), seconds (ss) and tenths and hundredths of seconds (th). A delay time interval may be specified using a REXX variable, coded with a leading ampersand, that contains the interval in the format, hhmmssth.

PERCENT

An optional keyword, and specifies that delay times specified by DELAY commands subsequently executed by execs that run on the CA Enterprise Tester port on which a DELAY PERCENT command is executed, are to be factored by the specified percent.

percent

May be an integer value from zero to 999, or may be a variable field that contains an integer from zero to 999, and is the percent of specified delay times to be used when DELAY commands are subsequently executed.

DELAY commands created by recording a session as a REXX exec capture the elapsed time, often called think time, that it took for a user to press enter or another key that sends data, after receiving the preceding message from the online system. When a DELAY command is executed, exec execution is delayed until the recorded or specified delay time elapses, simulating the think time delays that occur when people use terminals, and causing a session produced by a recorded exec to proceed at the same speed as the captured session.

The speed of execs that issue DELAY commands may be controlled using a DELAY command that specifies a delay PERCENT. When a DELAY PERCENT command is executed that specifies a percent less than 100, exec execution is delayed only for a percent of the time specified by each subsequent DELAY command, and the exec runs faster. When a percent greater than 100 is specified, delays are longer than the delay times specified by subsequent DELAY commands, causing the exec to run slower.

Because DELAY PERCENT applies to all execs that run on a CA Enterprise Tester port, an exec that calls other execs can control the speed of the called execs by issuing a DELAY PERCENT command.

At the completion of a DELAY command, updated information is returned in session variables that reflects any outbound message(s) received during a delay. Refer to CA Enterprise Tester REXX Variables on page 12-3 for a description of the information returned in session variables.

Conditions and Return Codes

At the completion of a DELAY command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. The following table describes DELAY command return codes and REXX conditions that are raised by default when some CA Enterprise Tester conditions occur.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
LOCKOUT	4		Timed out waiting for port lock
TIMEOUT	8		Timed out waiting for response
LOGOFF	16		Session Ended
ENDTRACE	20	HALT	Trace abended (ie., SD37)
CANCEL	24	HALT	CANCEL command processed
FAULT	64	HALT	Invalid command or system error

DELETEQ

The DELETEQ command is used to delete the entire data queue that is anchored to the port being accessed by the issuing exec.

Example

```
" DELETEQ     "
```

Format

```
  DELETEQ
```

Conditions and Return Codes

At the completion of a DELETEQ command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. The following table describes DELETEQ command return codes and default REXX conditions that may be raised for some CA Enterprise Tester conditions.

Conditions	RC	Default Signal	Reason for Condition
LOCKOUT	0		Timed out waiting for port lock
CANCEL	24	HALT	CANCEL command processed
FAULT	64	HALT	Invalid command or system error

ENDTRACE

The ENDTRACE command ends a session trace anchored to the port being accessed by an exec when the command is issued. A session trace is started by the TRACE command.

Example

```
" ENDTRACE    "
```

Format

```
   ENDTRACE
```

Conditions and Return Codes

At the completion of an ENDTRACE command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. The following table describes ENDTRACE command return codes and default REXX conditions that may be raised for some CA Enterprise Tester conditions.

Conditions	RC	Default Signal	Reason for Condition
LOCKOUT	0		Timed out waiting for port lock
CANCEL	24	HALT	CANCEL command processed
FAULT	64	HALT	Invalid command or system error

HANDLE

The HANDLE command activates or inactivates automatic handling of named session events when they occur during the processing of subsequently executed CA Enterprise Tester session commands.

Examples

```
" HANDLE turnaround off    "" HANDLE outbound off      "
```

Format

```
HANDLE event option
```

event

One of the keywords, TURNAROUND or OUTBOUND, that names the session event for which handling is being specified.

Option

One of the keywords, ON or OFF, that specifies whether automatic handling of the named event is to be activated or inactivated.

The HANDLE command TURNAROUND keyword refers to the point in a session that occurs when a TYPE command has sent input to an online system and changes the session direction to outbound. By default or when HANDLE TURNAROUND ON is specified, the TYPE command automatically waits for and receives the online system response to a terminal input. When HANDLE TURNAROUND OFF is specified, subsequent TYPE commands do not wait for the online system response but rather complete immediately after the session direction is changed to outbound. HANDLE TURNAROUND OFF makes the TYPE command a send-only operation. After executing a TYPE command while TURNAROUND handling is off, an exec must eventually use the INVITE command to receive the online system response to the input sent by a TYPE command.

The HANDLE command OUTBOUND keyword refers to the point in a session at which an outbound message has been received from an online system that does not free the terminal keyboard, but rather leaves the session in the outbound state. By default or when HANDLE OUTBOUND ON is specified, when a CA Enterprise Tester session command receives a message from an an online system that leaves the session in the outbound state, then CA Enterprise Tester automatically waits for and receives the next message sent by the online system. When HANDLE OUTBOUND OFF is specified, TYPE and INVITE commands that receive messages that leave the session in the outbound state, complete following the receipt of each message, whether or not the terminal keyboard is freed. After executing a TYPE or INVITE command while OUTBOUND handling is off, that ends when a message is received that leaves the session in an outbound state, an exec must eventually use the INVITE command to receive the next message sent by the online system.

When HANDLE commands are used to obtain control of outbound messages, and when TYPE and INVITE commands complete while the virtual terminal keyboard is not free, an exec must be aware of the state of a session and act accordingly. Specifically, a TYPE command is valid only while the keyboard is free. An INVITE command must eventually be issued to receive an online system message while the keyboard is locked. Execs that choose to handle outbound message flows may use CA Enterprise Tester-defined variable fields, PTEKB or PTESTATE, to determine the state of a session and to decide whether TYPE commands are valid or INVITE commands are required. Refer to CA Enterprise Tester REXX Variables for descriptions of PTEKB or PTESTATE and the possible values of the variable fields.

Conditions and Return Codes

At the completion of a HANDLE command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the HANDLE command. Table 11-7 describes HANDLE command return codes and default REXX conditions that may be raised for some CA Enterprise Tester conditions.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
CANCEL	24	HALT	CANCEL command processed
FAULT	64	HALT	Invalid command or system error

INVITE

The INVITE command receives any message sent by an online system within a specified time interval, and may be used to receive a message that is sent by an online system while the terminal keyboard is free, or may be used to receive messages when the state of a session is outbound after using HANDLE OUTBOUND, HANDLE TURNAROUND, or WAIT time ON RESPONSE commands.

Examples

```
" INVITE 1000 "" INVITE &invtime "
```

Format

```
INVITE hhmmssth
```

hhmmssth

An optional time interval, expressed in hours (hh), minutes (mm), seconds (ss) and tenths and hundredths of seconds (th).

The INVITE command is used to receive a message at a virtual terminal that is anchored to the port that is being accessed by an exec. The LOGON command allocates and anchors a virtual terminal to an accessed port and uses the terminal to start a session.

A CA Enterprise Tester LOGON command completes immediately after a session is started. The virtual terminal keyboard is always free at the beginning of a session, as is the keyboard on a real terminal. While it is true that a terminal keyboard is free before the first message is sent on a new session, this is seldom noticed by a person. After starting a session, most online systems send a message to the terminal. People are naturally patient and wait until they see the expected message.

REXX execs that use CA Enterprise Tester host commands to start and use terminal sessions with an online system, take the place of a terminal operator and must do the things that a person would do if a session is to flow smoothly. The INVITE command allows an exec to wait for an expected message, as a person would. At the start of a session, if an online system sends one or more initial messages, an exec must execute one or more INVITE commands to allow the online system to send the messages.

After receiving a terminal input, some online systems and applications send a message that frees the terminal keyboard immediately after receiving the terminal input. Then, at a later time, the online system sends another message that a terminal user sees as the response to the terminal input. Like a signon prompt sent at the start of a session, the unnoticed free keyboard situation does not cause a problem because the terminal operator is patient and waits for the expected message. A TYPE command ends when an online system message frees the terminal keyboard. If an online system sends a response that frees the terminal keyboard before sending the response message the user is expecting, then an exec must add patience by executing INVITE commands whenever it is necessary to wait for the delayed response.

Because people do not notice some free keyboard situations, it is sometimes difficult to determine when to code INVITE commands. One way todetermine if an INVITE is needed is to examine a CA Enterprise Tester trace of a session with the online system or application in question. A trace may be created using the Interactive System by first starting a trace, then starting a session on the same port, and then using the session to enter functions and/or data.

The top border of each traced screen contains an expression that shows the direction of the message that was being sent when the screen image was captured. If a screen image that documents a message that was sent from the online system to a terminal, contains an EB indicator, (an SNA End Bracket indicator), and the next traced screen image was also sent from the online system to the terminal, then an INVITE statement should be used in an exec that uses the traced application, to receive the second outbound message.

At the completion of an INVITE command, updated information is returned in session variables, as documented in CA Enterprise Tester REXX Variables.

Conditions and Return Codes:

At the completion of an INVITE command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. The following table describes INVITE command return codes and default REXX conditions that may be raised for some CA Enterprise Tester conditions.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
LOCKOUT	4		Timed out waiting for port lock
TIMEOUT	8		Timed out waiting for response
LOGOFF	16		Session ended
ENDTRACE	20	HALT	Trace abended (ie., SD37)
CANCEL	24	HALT	CANCEL command processed
FAULT	64	HALT	Invalid command or system error

LOCK

The LOCK command locks the port that is being accessed by an exec and all resources anchored to the port.

The ACCESS command may also be used to lock a port when it is accessed.

While a port is locked, resources anchored to the port may be used only by execs that run on same port as the exec that issued the LOCK command. If an exec running on any other port issues a CA Enterprise Tester command that uses resources anchored to the locked port, the command is delayed until the port is unlocked.

The UNLOCK command is used to unlock a locked port.

Example

```
" LOCK  "
```

Format

```
LOCK
```

Note: Refer to Sharing Resources with Attached Execs for more information about locks and uses of locks.

Conditions and Return Codes

At the completion of an LOCK command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. The following table describes LOCK command return codes and default REXX conditions that may be raised for some CA Enterprise Tester conditions.

Conditions	RC	Default Signal	Reason for Condition
LOCKOUT	0		Timed out waiting for port lock
CANCEL	24	HALT	CANCEL command processed
FAULT	64	HALT	Invalid command or system error

LOGOFF

The LOGOFF command ends a session that was started using the LOGON command.

Example

```
" LOGOFF "
```

Format

```
LOGOFF
```

The LOGOFF command ends the session anchored to the port being accessed by an exec when the command is executed.

When the LOGOFF command is used to end a session, the online system sees the session termination in exactly the same way that it sees a session termination that occurs when a user presses the the SYSREQ key on a real terminal keyboard, then enters the command LOGOFF, to request that VTAM terminate the active application session.

The preferred way to end a session is to issue a TYPE command that enters a particular online system's logoff command or transaction. Entering a logoff transaction may be required for some online systems to give the online system an opportunity to clean up resources that were allocated for the user signed on.

At the completion of a LOGOFF command, all session variables described in CA Enterprise Tester REXX Variables are dropped.

Conditions and Return Codes

At the completion of a LOGOFF command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. The following table describes LOGOFF command return codes and default REXX conditions that may be raised for some CA Enterprise Tester conditions.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
LOCKOUT	4		Timed out waiting for port lock
ENDTRACE	20	HALT	Trace abended (ie., SD37)
CANCEL	24	HALT	CANCEL command processed
FAULT	64	HALT	Invalid command or system error

LOGON

The LOGON command allocates a CA Enterprise Tester virtual terminal, anchors the terminal to the port being accessed by the exec when the LOGON command is executed, and uses the terminal to start a session with a named online system.

Examples

```
" LOGON cics1 model mod2 "" LOGON &system data &uid  "
```

Format

```
LOGON system-name (MODEL terminal-modelMOD2 | MOD3 | MOD4 | MOD5 ) ( default-
screen-size24X80 | 32X80 | 43X80 | 27X132 ) ( alternate-screen-size24X80 | 32X80
| 43X80 | 27X132 )| ( terminal-type

  BASICDS | EXTDS )

 (FROM terminal-name )

 (DATA logon-data )

 ( buffer-format

BASICDATA | EXTDSDATA | PANELDATA | IMAGEDATA |  ATTRIBUTES attr-values )(
USERMOD ispf-usermod )
```

system-name

The 1 through 8 character name of an online system with which a session is to be started. The system name specified may be an actual name, a literal value or a REXX variable, coded with a leading ampersand, that contains the name of an online system.

MODEL

An optional keyword and indicates that the value that follows is a symbolic terminal model name or the name of a mode table entry.

terminal-model

The logon mode table entry to be used to start a session, and may be one of the keywords MOD2, MOD3, MOD4 or MOD5, to refer to a mode table entry name or may be the actual name of a logon mode table entry. terminal-model may be expressed as an actual value, or may be a variable field that contains a valid terminal model value or null value.

default-screen-size

Optional, and is the default virtual terminal screen size to be used for a session. default-screen-size may be one of the keywords 24X80, 32X80, 43X80 or 27X132; and may be expressed as an actual value, or may be a variable field that contains a screen size keyword or a null value.

alternate-screen-size

Optional, and is the alternate virtual terminal screen size to be used for a session. alternate-screen-size may be one of the keywords 24X80, 32X80, 43X80 or 27X132; and may be expressed as an actual value, or may be a variable field that contains a screen size keyword or a null value;

terminal-type

Optional, and specifies the type of terminal to be used for a session. terminal-type may be one of the keywords BASICDS or EXTDS, and may be expressed as an actual value, or may be a variable field that contains a terminal type keyword or null value.

FROM

An optional keyword and indicates that the data item that follows is a virtual terminal name.

terminal-name

The name of the virtual terminal to be used for the session. terminal-name may be the actual name of a VTAM APPL that has been defined at an installation for use as a CA Enterprise Tester virtual terminal or may be a variable field that contains a virtual terminal name or a null value. If the FROM keyword is not specified or if terminal-name is a variable field that contains a null value, then a virtual terminal will be assigned automatically.

DATA

An optional keyword and indicates that the data item that follows is logon data.

logon-data

A data item identified by the DATA keyword and is logon data that is to be sent to an online system with a session request. logon-data may be a variable field that contains logon data and may be up to 255 characters in length; or may be one or more consecutive literal values whose accumulated length(s) do not exceed 255 characters. When the DATA keyword is not specified or if logon-data is a variable field that contains no data, then data is not included in the session request sent to an online system.

buffer-format

An optional keyword that specifies the format of the virtual terminal display buffer data returned in REXX variables, PTEBUFF and PTEEAB, at the completion of the LOGON command and subsequent session commands. Refer to 3270 Attribute Values for the values and meanings of basic, panel and extds attributes and extended attribute data.

- By default, basic field attribute values are returned in PTEBUFF and extended attribute data is returned in PTEEAB only if an EXTDS session is active.
- BASICDATA returns basic attributes in PTEBUFF and does not return PTEEAB, even if an EXTDS session is active.
- EXTDSDATA returns extds attributes in PTEBUFF and returns extended attribute data in PTEEAB, even if a BASICDS session is active.
- PANELDATA returns panel attributes in PTEBUFF, blanks out protected non-display data and translates nulls in protected display fields to blanks. PTEEAB is not returned.
- IMAGEDATA translates all non-display data, attributes and and nulls to blanks in the returned PTEBUFF, and does not return PTEEAB. IMAGEDATA produces a legible PTEBUFF that may be printed or used in application documentation.
- ATTRIBUTES may be used to specify the 32 1-byte hexadecimal values to be returned as the 32 possible basic 3270 field attributes in PTEBUFF. The value following ATTRIBUTES, attribute-values, is a 64-character literal value enclosed in single or double quotes. Each character is a hexadecimal character, 0-F,and each successive pair of characters is a hexadecimal value to be used to represent the next successive basic field attribute. When ATTRIBUTES is specified, PTEEAB is returned if an EXTDS session is active.

USERMOD

An optional keyword and indicates that the value that follows is an ISPF usermod value.

ISPF-usermod

Immediately follows the USERMOD keyword and specifies a one-byte value, expressed as a single character or as a two-character hexadecimal value enclosed in single or double quotes. The ISPF-usermod value tells CA Enterprise Tester the USERMOD value specified by an ISPF panel that is used to display an image of a virtual terminal screen. ISPF-usermod defaults to hexadecimal 3F.

The system name specified may be the actual VTAM APPL name by which an online system is known to VTAM or may be a familiar system id defined in CA Enterprise Tester system options, that refers to an actual VTAM APPL. Defined system ids may be displayed on the Interactive System System Menu.

If a null terminal model is specified in a CA Enterprise Tester LOGON command, or if the MODEL keyword is not specified, then a default mode table entry name specified during CA Enterprise Tester installation is used. The following table lists terminal model keywords and their associated default and alternate virtual terminal screen sizes.

Model Keyword	Default Size	Alternate size
MOD2	24 X 80	24 X 80
MOD3	24 X 80	32 X 80
MOD4	24 X 80	43 x 80
MOD5	24 X 80	27 x 132

The first screen size keyword encountered in a LOGON command specifies the default screen size. If a default screen size is specified, it is used only when a session is started using a mode table entry that does not contain screen sizes, and sizes are not forced by an online system.

The second screen size keyword encountered in a LOGON command specifies the alternate screen size. If an alternate screen size is specified, it is used only when a session is started using a mode table entry that does not contain screen sizes, and sizes are not forced by an online system.

If screen sizes are not specified and are not defined in a the mode table used to start a session and are not forced by an online system, then default sizes specified at an installation are used.

A terminal may be either a basic or extended 3270 data stream terminal. A basic data stream terminal has a single screen buffer that is used to store screen data and coded basic field attributes. An extended data stream terminal has a screen buffer that is used to store screen data and actual basic field attributes and has an extended attribute buffer that is used to store extended data stream color and highlighting attributes and other information.

If a terminal type is not specified, a default type specified at an installation is used. If an online system sends a 3270 Read Partition Query structured field to a terminal to determine device capabilities, the terminal type or screen sizes specified influence the contents of the reply.

After using the LOGON command to start a session, the CA Enterprise Tester TYPE, INVITE, and DELAY commands are used to key data into the virtual terminal used for a session, to send input to the online system, to receive online system responses to input, and to receive messages initiated by an online system.

ISPF Session Panels explains how to use the LOGON command PANELDATA and USERMOD options to display an image of a virtual terminal screen using an ISPF panel, and how to key data entered by the user into a virtual terminal using the TYPE command USERMOD option.

At the completion of each CA Enterprise Tester session command, session information is returned to the issuing exec in REXX variables. Refer to CA Enterprise Tester REXX Variables for a description of each session variable.

A session is ended by an online system when a logoff transaction is entered using a TYPE command, or may be ended using a LOGOFF command. Sessions are terminated automatically if all execs in an address space end and leave sessions active.

Conditions and Return Codes

At the completion of a LOGON command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. The following table describes LOGON command return codes and default REXX conditions that may be raised for some CA Enterprise Tester conditions.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
LOCKOUT	4		Timed out waiting for port lock
LOGOFF	16		Session ended
ENDTRACE	20	HALT	Trace abended (ie., SD37)
CANCEL	24	HALT	CANCEL command processed
FAULT	64	HALT	Invalid command or system error

MONITOR

The MONITOR command instructs CA Enterprise Tester to monitor or to stop monitoring specified events and specifies what to do when monitored events are detected. Detected events may be displayed at a terminal, written as REXX SAY messages, or may passed to a user-written exec that may process the event as desired. MONITOR command instructions apply to subsequent CA Enterprise Tester commands issued by any exec that runs in the REXX Language Processor Environment in which the MONITOR command is issued, including execs that call or that are called by an exec that issues a MONITOR command. A MONITOR command may instruct CA Enterprise Tester to monitor or to stop monitoring:

- CA Enterprise Tester commands issued by REXX execs; and/or
- online system responses or other messages received by CA Enterprise Tester session commands; and/or
- REXX variable values assigned or used by CA Enterprise Tester commands; and/or
- REXX conditions raised by CA Enterprise Tester commands to indicate CA Enterprise Tester conditions to an exec; and/or
- CA Enterprise Tester conditions returned to an exec as a return code.

Examples

```
" MONITOR SIGNALS "

" MONITOR SIGNALS CONDITIONS COMMANDS RESPONSES STORE DROP FETCH "

" MONITOR &REXXSYM &PTEMESS "

" MONITOR SIGNALS CONDITIONS COMMANDS RESPONSES VIAEXEC MONEXEC "

" MONITOR OFF "
```

Format

```
MONITOR ( events )( &generic_symbols )( OFF )( VIASAY | VIATPUT | VIAEXEC
monitor_exec )( NOTE &note | RECALL &note )
```

events

An optional list of keywords that name events to be monitored. When OFF is also specified monitoring is stopped for the specified events. Event keywords are:

- SIGNALS monitors REXX HALT, ERROR or FAILURE conditions raised by CA Enterprise Tester commands to indicate CA Enterprise Tester conditions to an exec.

- CONDITIONS monitors return codes returned to execs by CA Enterprise Tester commands to indicate CA Enterprise Tester LOCKOUT, TIMEOUT, ENDQUEUE, LOGOFF, ENDTRACE, CANCEL or FAULT conditions.
- COMMANDS monitors the start of subsequent commands.
- RESPONSES monitors online system responses and other messages sent by an online system to a virtual terminal, and session initiation and termination events, when they are received by CA Enterprise Tester session commands. When RESPONSES are received, information is not displayed; instead a specified or default monitor_exec is called to process the information. The distributed default monitor_exec, PTEMONX, shown in The Distributed Monitor Exec on page 11-29 displays commands and virtual terminal screen images using ISPF display services or the REXX SAY instruction, depending on the REXX environment.
- STORE monitors changes to variables by CA Enterprise Tester commands or monitors changes to variable values whose names begin with a specified &generic_symbols.
- FETCH monitors uses of variables by CA Enterprise Tester commands or monitors uses of variables whose names begin with a specified &generic_symbols.
- DROP monitors variables dropped by CA Enterprise Tester commands or monitors dropped variables whose names begin with a specified &generic_symbols.

&generic_symbols

Monitors operations performed by CA Enterprise Tester commands on REXX variables whose names begin with any specified generic_symbol, or stops monitoring when OFFis also specified. For example, MONITOR &PTE &Q_VALUE, monitors CA Enterprise Tester operations on variables whose names begin with PTEor Q_VALUE. When STORE, FETCH and/or DROPare also specified, only the specified operations are detected, otherwise all operations on variables whose names begin with specified &generic_symbols are detected.

OFF

Stops monitoring for specified events and/or &generic_symbols. If OFF is the only MONITORcommand option specified, then all monitoring is stopped.

VIASAY

Causes messages to be sent to the REXX SAY destination when monitored events other than RESPONSES are detected.

VIATPUT

Causes messages to be sent to the TSO terminal using TSO TPUT, when monitored events other than RESPONSESare detected. In a non-TSO address space, VIATPUT sends messages to the REXX SAY destination.

VIAEXEC monitor_exec

Specifies that a named exec, monitor_exec, is to be called when a monitored event is detected. The Monitor Exec Interface on page 11-28 documents arguments passed to a monitor_exec when it is called to process events, and provides other information about monitor execs.

NOTE &note

Saves the value of REXX variable, note for subsequent recall by any exec that runs on the same port as the exec that saved the &note.

RECALL &note

Returns a monitor note in REXX variable note that was saved by a MONITOR NOTE command issued by any exec running on the same port as the exec that recalls the &note.

By default, CA Enterprise Tester commands are not monitored. While return codes, REXX conditions and information returned in REXX variables informs an exec about errors or other CA Enterprise Tester conditions encountered by CA Enterprise Tester commands, no information about CA Enterprise Tester conditions is automatically displayed or otherwise made available to the user of an exec.

- A MONITOR command that specifies only SIGNALS may be used to display or SAY CA Enterprise Tester error messages automatically.
- A MONITOR command that specifies RESPONSES may be added to a recorded or user written exec that uses CA Enterprise Tester session commands to automatically display the session while the exec is executing.
- MONITOR commands that specify other events and/or variables may be used to perform more extensive exec debugging.

Because MONITOR commands apply to all execs that run in a REXX environment, general purpose execs that activate monitoring and call other execs may be used to debug execs without modification to the exec being debugged. In addition to debugging applications, the MONITOR command, monitor execs, other CA Enterprise Tester commands, and ISPF commands may be used to monitor multi-session stress tests. More information about and examples of MONITOR command applications may be found in Monitoring Execs and Sessions.

Conditions and Return Codes

At the completion of a MONITOR command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. The following table describes MONITOR command return codes and default REXX conditions that may be raised for some CA Enterprise Tester conditions.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
CANCEL	24	HALT	CANCEL command processed
FAULT	64	HALT	Invalid command or system error

The Monitor Exec Interface

A monitor exec specified using the MONITOR command VIAEXEC option is called to process all monitored events when they are detected. A default monitor exec is called to process online system responses when RESPONSES are monitored, but a monitor exec has not been specified using the MONITOR command VIAEXEC option. When a monitor exec is called to process an event, information about the event is passed to the monitor exec as CALL arguments. A monitor exec may assign the argument values to REXX variables using a REXX ARG instruction that has commas coded between specified variable names. The CALL arguments passed to a monitor exec are:

- The name of the REXX Environment in which the monitored event was detected, as specified by REXX parameters when the environment was initialized, possibly z/OS, TSO/E or ISPF.
- The name of the REXX exec that issued the CA Enterprise Tester command that was being processed when the monitored event was detected.
- The text of the CA Enterprise Tester command being executed when the monitored event was detected.
- One of the keywords, COMMAND, RESPONSE, STORE, FETCH, DROP or CC, that identifies the event to be processed by the monitor exec.
- When argument 4 is STORE, FETCH or DROP, then argument 5 is the name of the REXX variable whose value was stored, fetched or dropped.
- When argument 4 is CC, then argument 5 is one of REXX conditions HALT, ERROR or FAILURE, if a REXX condition is being raised by a CA Enterprise Tester command, or is RC if a return code is being returned to an exec to indicate a CA Enterprise Tester condition.
- When argument 4 is STORE or FETCH, then argument 6 is the value of the REXX variable that was stored or fetched.

- When argument 4 is CC, then argument 6 is one of the CA Enterprise Tester condition keywords, LOCKOUT, TIMEOUT, ENDQUEUE, LOGOFF, ENDTRACE, CANCEL or FAULT, and indicates why a REXX condition is being raised or a return code is being returned to an exec.
- When argument 4 is CC, argument 7 is a CA Enterprise Tester message that further documents why a REXX condition is being raised or a return code is being returned to an exec.

Prior to calling a monitor exec, active monitoring is turned off, and monitoring is restored when a monitor exec returns. A monitor exec may issue an ACCESS LASTPORT command to determine the port being accessed by the exec when the monitored event was detected, may issue a QUERY SESSION command to fetch a copy of all CA Enterprise Tester defined session variables, or may issue any other CA Enterprise Tester command including the MONITOR command. Monitor execs may also ADDRESS other command processors and use other commands that are valid in the REXX Language Processor Environment.

MONITOR NOTE and MONITOR RECALL functions are intended for use by monitor execs and other execs that control monitor functions and monitor execs. An exec that issues a MONITOR command that specifies a user-written monitor exec may pass information to the monitor exec using a monitor note. A monitor exec may recall information noted by another exec or may recall, update and note session statistics or other information needed when the monitor exec is called to process subsequent events. While there are no restrictions regarding the use of the MONITOR NOTE or RECALL functions, it is recommended that the functions be used only by monitor execs or execs that communicate or cooperate with monitor execs, to avoid conflicting use of notes if monitor execs are used to monitor other execs.

The Distributed Monitor Exec

If a MONITOR command specifies RESPONSES, but does not specify VIAEXEC, a default monitor exec specified in CA Enterprise Tester system options is called when subsequent online system responses or other outbound session events are processed by CA Enterprise Tester session commands.

A distributed monitor exec, PTEMONX is designed to be used as a default monitor exec. When called to process detected RESPONSE events, PTEMONX, displays CA Enterprise Tester session commands and virtual terminal screen images using ISPF display and pop-up services, when called in an ISPF REXX Language Processor Environment, or using the REXX SAY instruction when called in a TSO or other REXX Language Processor Environment.

PTEMONX is distributed in the CA Enterprise Tester EXEC library. ISPF panels used by PTEMONX, PTEPXMDT and PTEPXMW2 are distributed in the CA Enterprise Tester PLIB library.

PTEMONX should not be specified as a MONITOR command VIAEXEC. If called to process an event other than a RESPONSE event, PTEMONX ignores the event and returns. Monitored events other than RESPONSE events are displayed automatically if a VIAEXEC is not specified. However, a user-written monitor exec that processes other events may call PTEMONX to process RESPONSE events provided that it passes all arguments passed to it by CA Enterprise Tester.

The following is an example of the PTEMONX default monitor exec:

```
//* PTEMONX - A Default ProTerm Monitor Exec for REXX (distributed) */
arg p_spc, p_exe, p_cmd, p_eve, p_sig, p_cnd, p_msg
if p_eve \= 'RESPONSE' then exit
parse upper var p_exe execname . ; parse upper var p_cmd verb .
address ProTerm " access lastport "
if p_spc = 'ISPF' then do
green = '01'x ; red = '05'x ; blue = '11'x ; white = '15'x
address ProTerm " query session paneldata "
address ispexec
zwinttl = execname 'executed command on Port' PTEAPORT
if PTESTATE.PTEAPORT \= 'PTESTATE.'PTEAPORT then do
if verb = 'LOGON' then do
PTEDESKT = ''
" control nondispl " ; " display panel(PTEPXMDT) "
PTEMWIN2 = left(white || p_cmd,72) || right(blue,
|| 'Result:' || green || 'Session started with',
PTEPLU.PTEAPORT,72)
end
else PTEMWIN2 = left(white || p_cmd,72),
|| right(green || 'Response follows . . .',72)
" addpop row(19) column(1) "
" control display lock " ; " display panel(PTEPXMW2) "
address ProTerm " wait 500 "
" rempop "
if verb \= 'LOGON' then do
if PTECOLS.PTEAPORT = 80 then
PTEDESKT = PTEBUFF.PTEAPORT
else do
PTEDESKT = '' ; pos = 1
do row = 1 to PTEROWS.PTEAPORT
PTEDESKT = PTEDESKT,
|| substr(PTEBUFF.PTEAPORT,pos,80)
pos = pos + PTECOLS.PTEAPORT
end
end
" control display lock "
" display panel(PTEPXMDT) ",
" cursor(PTEDESKT) csrpos("PTECSRP.PTEAPORT")"
address ProTerm " wait 500 "
end
end
else do
PTEMWIN2 = left(white || p_cmd,72) || right(blue,
|| 'Result:' || red || 'Session Ended.',72)
" addpop row(19) column(1) "
" control display lock " ; " display panel(PTEPXMW2) "
address ProTerm " WAIT 500 "
" rempop "
end
end
else do
address ProTerm " QUERY SESSION IMAGEDATA "
say ' ' ; say execname 'executed command on Port' PTEAPORT':'
say ' ' p_cmd
if PTESTATE.PTEAPORT \= 'PTESTATE.'PTEAPORT then do
if verb = 'LOGON' then
say = 'Result: Session started with' PTEPLU.PTEAPORT
else do
say 'Response follows . . .' ; say copies('_',79)
pos = 2
do row = 1 to PTEROWS.PTEAPORT
say substr(PTEBUFF.PTEAPORT,pos,79)
pos = pos + PTECOLS.PTEAPORT
end
end
say copies('_',79)
end
else say 'Result: Session Ended'
end
```

POST

The POST command replaces the entire queue anchored to the port being accessed by an exec when the command is issued, with a single queue entry that contains specified argument values. If a PULL command issued by an exec running on another port is waiting for data to be stored in the queue, then the oldest waiting PULL command is resumed when the queue entry is posted.

Examples

```
" POST 'saymsg' &PTEMESS "
" POST &item1 &item2 &item3 "
```

Format

```
 POST (arg1 . . . arg20)
```

arg1 through arg20

Optional and specify up to 20 argument values, either literal values contained in single or double quotes, or REXX variables, coded with a leading ampersand, that contain argument values to be stored in the posted queue entry. A null queue entry is posted if no arguments are specified.

If the POST command is used to post updated information and QUERY QUEUE commands are used to examine the information, a queue becomes a bulletin board that may be used to provide current information to any number of other execs. QUEUEand PULL commands may be used to process queue entries in the order that they were queued. PUSH and PULL commands may be used to to process the most recently queued entry first.

At the completion of a POST command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
LOCKOUT	4		Times out waiting for port lock
CANCEL	24	HALT	CANCEL command processed
FAULT	34	HALT	Invalid command or system error

PULL

The PULL command assigns argument values fetched from the top entry of the queue anchored to the port being accessed by an exec when the command is issued, to variables specified by the PULL command.

Example

```
" PULL &var1 &var2 &var3 "
```

Format

```
PULL (&variable1 . . . &variable20)
```

&variable1 through &variable20

The names of up to 20 variables, coded with a leading ampersand, to which pulled argument values may be assigned.

Argument values stored in the pulled queue entry by a QUEUE, PUSHor POST command are returned in the corresponding variables specified by a PULL command.

If the number of variables specified exceeds the number of argument values in the pulled queue entry, then a zero-length value is assigned to excess variables. If the number of argument values in a pulled queue entry exceeds the number of variables specified by the PULL command, then excess argument values are discarded.

A CA Enterprise Tester WAIT command that specifies ON PULL may be issued to specify a time interval that subsequent PULL commands are to wait for data to be queued in the event that an accessed queue is empty when a PULL command is issued. A PULL completes normally if queued data is immediately available or if data is queued within a specified time interval. If an accessed queue is empty and a WAIT interval was not specified or a specified interval expires before data is queued, the PULL command completes with return code 12 to indicate the CA Enterprise Tester ENDQUEUE condition.

Conditions and Return Codes

At the completion of a PULL command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. The following table describes PULL command return codes and default REXX conditions that may be raised for some CA Enterprise Tester conditions.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
LOCKOUT	4		Times out waiting for port lock
ENDQUEUE	12		Data queue was empty
CANCEL	24	HALT	CANCEL command processed
FAULT	34	HALT	Invalid command or system error

PUSH

The PUSH command adds an entry to the top of the queue anchored to the port being accessed by an exec when the command is issued, that contains specified argument values. If a PULL command issued by an exec running on another port, is waiting for data to be stored in the queue, then the oldest waiting PULL command is resumed when an entry is queued by a PUSH command.

Examples

```
" PUSH 'INQUIRE' &ACCOUNT  "
```

```
" PUSH &item1 &item2 &item3 "
```

Format

```
PUSH (arg1 . . . arg20)
```

arg1 through arg20

Optional and specify up to 20 argument values, either literal values contained in single or double quotes or REXX variables, coded with a leading ampersand, that contain argument values to be stored in the queue entry. A null entry is queued if no arguments are specified.

When the PUSH command is used to queue data and the PULL command is used to process it, the most recently queued entries are processed first. QUEUE and PULL commands may be used to to process queue entries in the order that they were queued. A queue may also be used as a bulletin board to post current information that may be examined to any number of other execs using the POST and QUERY QUEUE commands.

At the completion of a PUSH command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. The following table describes PUSH command return codes and default REXX conditions that may be raised for some CA Enterprise Tester conditions.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
LOCKOUT	4		Times out waiting for port lock
CANCEL	24	HALT	CANCEL command processed
FAULT	34	HALT	Invalid command or system error

QUERY

The QUERY command returns requested information about CA Enterprise Tester resources in REXX variables. The QUERY command may be used to:

- Obtain information about the CA Enterprise Tester session, session trace, data queue or exec task on the port being accessed by the interrogating exec.
- Obtain a list of ports that have active CA Enterprise Tester sessions, active session traces, queued data, or active REXX exec tasks.

Examples

```
" QUERY session          "

" QUERY sessions         "

" QUERY queue &var1 &var2 &var3 "
```

Format

```
QUERY
    ( QUEUE (&variable1 . . . &variable20) | QUEUES | SESSION ( LOGONDATA |
   BASICDATA | EXTDSDATA |
   PANELDATA | IMAGEDATA ) | SESSIONS | TASK | TASKS |
  TRACE | TRACES )
```

QUEUE, QUEUES, SESSION, SESSIONS, TASK, TASKS, TRACE and TRACES

Keywords that identify the CA Enterprise Tester resource or resources about which information is to be returned.

&variable1 . . . &variable20

Can be specified when a QUEUE is queried, and are the names of up to 20 REXX variable names, coded with a leading ampersand, that the queried argument values are to be assigned.

LOGONDATA, BASICDATA, EXTDSDATA, PANELDATA and IMAGEDATA

Keywords that may be specified when a SESSIONis queried to specify the format of the virtual terminal display buffers returned in variables, PTEBUFFand PTEEAB.

When QUEUES, SESSIONS, TASKS or TRACES is specified, NOQUEUES, NOSESSIONS, NOTASKS or NOTRACES is returned in variable, PTEINFO, if the queried resource does not exist on any CA Enterprise Tester port, otherwise QUEUES, SESSIONS, TASKS or TRACES is returned in PTEINFO, and a list of the ports on which the queried resource exists is returned in PTEPORTS. For example:

- If only the queues on ports 1, 2, and 6 contain data, the command, query queues, returns QUEUES in PTEINFO and returns 1 2 6 in PTEPORTS.
- If all queues are empty, NOQUEUES is returned in PTEINFO.
- Query tasks, never returns NOTASKS in PTEINFO, because at least the exec that issued the QUERY command is active on a port.

When QUEUE, SESSION, TASK or TRACE is specified, NOQUEUE, NOSESSION, NOTASK or NOTRACE is returned in PTEINFO, if the queried resource is not active on the port being accessed by the exec when the QUERY command is issued, otherwise QUEUE, SESSION, TASK or TRACE is returned in PTEINFO.

When a QUERY QUEUE command specifies one or more variables, &variable1 through &variable20, and the accessed queue contains data, then values obtained from the top entry in the queue are returned in corresponding variables specified by the QUERY command. If the number of variables specified exceeds the number of values in the top queue entry, a zero-length value is returned in excess variables. The QUERY QUEUE command works like the PULL command, except that the QUERY command does not remove data from a queue.

A QUERY SESSION command returns the same information in session variables that is returned by all CA Enterprise Tester session commands, and drops session variables if a session is not active on the accessed port. However, a QUERY SESSION command may specify one of the keywords, LOGONDATA, BASICDATA, EXTDSDATA, PANELDATA or IMAGEDATA, to override the format of the virtual terminal display buffers returned by the QUERY command in PTEBUFF and PTEEAB. By default or when LOGONDATA is specified, buffer data is returned using the format specified by the LOGON command that started the queried session. Refer to LOGON for a description of the format of the buffer data returned when other optional format keywords are specified. Refer to CA Enterprise Tester REXX Variables for a discription of all session variables.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
LOCKOUT	4		Times out waiting for port lock
CANCEL	24	HALT	CANCEL command processed
FAULT	34	HALT	Invalid command or system error

QUEUE

The QUEUE command adds an entry to the bottom of the queue anchored to the port being accessed by an exec when the command is issued, that contains specified argument values. If a PULL command issued by an exec running on another port is waiting for data to be stored in the queue, then the oldest waiting PULL command is resumed when the queue entry is added.

Examples

```
" QUEUE 'INQUIRE' &ACCOUNT  "

" QUEUE &item1 &item2 &item3 "
```

Format

```
QUEUE (arg1 . . . arg20)
```

arg1 through arg20

Optional and specify up to 20 argument values, either literal values contained in single or double quotes or REXX variables, coded with a leading ampersand, that contain argument values to be stored in the queue entry. A null entry is queued if no arguments are specified.

When the QUEUE command is used to queue data and the PULL command is used to process it, queue entries are processed in the order they were queued. PUSH and PULL commands may be used to to process the most recently queued entry first. A queue may also be used as a bulletin board to post current information that may be examined to any number of other execs using the POST and QUERY QUEUE commands.

At the completion of a QUEUE command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. Table 11-18 describes QUEUE command return codes and default REXX conditions that may be raised for some CA Enterprise Tester conditions.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
LOCKOUT	4		Times out waiting for port lock
CANCEL	24	HALT	CANCEL command processed
FAULT	34	HALT	Invalid command or system error

SIGNAL

The SIGNAL command instructs subsequent CA Enterprise Tester commands to signal the occurrence of a specified CA Enterprise Tester condition by raising a REXX ERROR, FAILURE or HALT condition, or to raise no REXX condition if a specified CA Enterprise Tester condition is encountered.

Examples

```
"     SIGNAL OFF ON ENDTRACE  "

"     SIGNAL ERROR ON LOGOFF  "
Formath
```

SIGNAL REXX-cond ON ProTerm-cond

REXX-cond is one of the keywords, ERROR, FAILURE, HALT or OFF.

- ERROR, FAILURE or HALT specify the REXX condition to be raised if the specified ProTerm-cond occurs, and are also the REXX condition keywords used in a REXX SIGNAL or CALL instruction to trap the CA Enterprise Tester-cond.
- OFF specifies that no REXX condition is to be raised if the specified CA Enterprise Tester-cond occurs.

ON is required and identifies the ProTerm-cond that follows.

ProTerm-cond is one of the CA Enterprise Tester condition keywords, LOCKOUT, TIMEOUT, ENDQUEUE, LOGOFF, ENDTRACE, CANCEL or FAULT.

All unusual events and errors that can be detected by CA Enterprise Tester commands are categorized into named groups called ProTerm conditions. Each ProTerm condition is associated with a unique return code that is returned in the REXX RCvariable when a ProTerm condition is detected. The following table is a summary of all possible ProTerm condition and return codes, and default REXX conditions raised for some CA Enterprise Tester conditions.

The CA Enterprise Tester SIGNAL command is designed for use with REXX SIGNAL and CALL instructions to trap CA Enterprise Tester conditions using REXX condition traps, or may be used to avoid REXX conditions that are raised by default for some CA Enterprise Tester conditions.

For example, instead of checking the return code following each CA Enterprise Tester session command, to detect and process an unexpected session failure, a CA Enterprise Tester SIGNAL command and a REXX SIGNAL instruction may be used to detect and process an unexpected session failure, without checking return codes.

If a session is traced, but a trace failure is not a big enough problem to interrupt an exec, the SIGNAL command may be used to override the REXX HALT condition that is raised by default when a CA Enterprise Tester ENDTRACE condition occurs.

```
"  SIGNAL OFF ON ENDTRACE
"
"
  trace
  to tracedd
"
"
  SIGNAL ERROR ON LOGOFF
"
  signal ON ERROR name ended
"
  logon
  cics
"
   say 'session ended:' PTEMESS
```

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
LOCKOUT	4		Timed out waiting for port lock
TIMEOUT	8		Timed out waiting for response
ENDQUEUE	12		Data queue was empty
LOGOFF	16		Session ended
ENDTRACE	20	HALT	Trace abended
CANCEL	24	HALT	CANCEL command processed
FAULT	34	HALT	Invalid command or system error

At the completion of a SIGNAL command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. The following table defines the CA Enterprise Tester conditions, return codes and default REXX conditions that may be raised by the SIGNAL command.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
CANCEL	24	HALT	CANCEL command processed
FAULT	34	HALT	Invalid command or system error

TRACE

The TRACE command starts a CA Enterprise Tester session trace on the port being accessed by an exec when the command is issued. While a session trace is active, CA Enterprise Tester writes the current image of a virtual terminal screen image to a trace data set whenever a TYPE command sends data to an online system and when an online system sends a response or other message to the virtual terminal being used for a session.

Examples

```
" TRACE to tracedd image freekb  "

" TRACE 'change caption to this' "

" TRACE to &ddname format pack "
```

Format

TRACE TO ddname

```
( FORMAT | IMAGE | UPPERCASE )
( DELAYS | DELAYSOFF )
( LOGONS | LOGONSOFF )
( CHAINS | FREEKB )
( PACK   | NOPACK )|
( caption )
```

TO ddname

Specifies the DD name to be opened to capture screen images when CA Enterprise Tester session commands are used to send and receive data using the session anchored to the same port as the session trace.

FORMAT

Optional and specifies that traced screen images are to contain all screen data. A FORMAT trace is required as input to session analyzers.

IMAGE

Optional and specifies that traced screen images are to be edited to convert attributes and other non-display data to blanks, and graphic escape characters to dashes, producing screen images similar to screen images visible to users of the system.

UPPERCASE

Optional and specifies that traced screen images are to be edited and converted to all upper case characters.

DELAYS

Specifies that user think time is to be captured as PSL DELAY statements in the trace data set. DELAYSOFF specifies that no DELAY statements are to be created.

LOGONS

Specifies that PSL LOGON statements are to be captured when a session is started. LOGONSOFF specifies that no LOGON statements are to be created.

CHAINS

Specifies that all outbound screen images are to be traced. FREEKB specifies that outbound screens images are to be captured only when an outbound message frees the terminal keyboard.

PACK

Specifies that traced session data is to be written using the ISPF packed data format. NOPACK specifies that traced screen images are to be written as unpacked, fixed-length records.

caption

a literal value that is to be used as the caption of subsequently traced screen images, and may be specified when a trace is started or at any time while a trace is active. A caption must be enclosed in either single or double quotes.

Default values for all optional parameters are controlled by CA Enterprise Tester system options.

The TRACE command starts or modifies the trace on the port that is being accessed TRACE command is issued.

A trace may be started or ended before, during and after a session is started or ended to capture any portion of a session.

A trace is ended using the ENDTRACE command or is ended automatically when all execs in an address space have ended.

The data captured when an exec-driven session is traced is similar to the data captured when a session is traced by other CA Enterprise Tester components. When a session is traced using the FORMAT option. the trace may be browsed, analyzed or converted to CA Enterprise Tester scripts, REXX execs, or image traces using CA Enterprise Tester Interactive System facilities.

At the completion of a TRACE command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. The following table describes TRACE command return codes and default REXX conditions that may be raised for some CA Enterprise Tester conditions.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
LOCKOUT	4		Times out waiting for port lock
ENDTRACE	20	HALT	Trace abended (ie., SD37)
CANCEL	24	HALT	CANCEL command processed
FAULT	34	HALT	Invalid command or system error

TYPE

The TYPE command operates a virtual terminal by pressing keys on a virtual terminal keyboard that perform the same functions as the keys on a real terminal keyboard. For example, a tab key moves the cursor to the next modifiable field. When a data key is pressed, each character of the data is keyed in at the current cursor position, the modified data tag is turned on in the attribute that controls the field, and the cursor advances one position or skips to the next field if the position is a skip attribute.

A TYPE command uses the keyboard of the virtual terminal that is anchored to the CA Enterprise Tester port being accessed by an exec when a TYPE command is issued, to position the cursor, to type literal, variable and other data values, to erase fields, to send data to an online system, and to perform other keyboard operations specified by the command. The keyboard operations specified by a TYPE command are performed in the order in which they are coded.

Examples

```
" TYPE home erase 'cemt set'  "," ' da(&dd) open ena' enter "

" TYPE clear  "

" TYPE <2 15> 'inq'  "

" TYPE <4 15> 'smith' enter "

" TYPE <&row &col> &data1  "" TYPE <4 15> &data2 enter "

" TYPE home tab &account "," tab &order pf2 "

" TYPE usermod <&cursor> "pfkey
```

Format

```
TYPE ( <row column> )
     ( <position> )
     ( HOME TAB BACKTAB NEWLINE )
     ( 'literal-data' )
     ( "literal-data" )
     ( &variable-data )
     ( ERASE )
     ( SELECT | DESELECT )
     ( TOKEN token )
     ( CODE code )
     ( USERMOD )
     ( ENTER | PF1-24 | PA1-3 |
     PEN | CLEAR | ATTN |AID &aid-variable )
     TYPE
```

<row column>

Moves the cursor to the specified virtual terminal screen row and column. A row or column may be a row or column number or the name of a REXX variable, coded with a leading ampersand, whose value is a row or column number.

<position>

Moves the cursor to the specified virtual terminal screen position, where; position = (row - 1) * PTECOLS + column. position may be a number or the name of a REXX variable, coded with a leading ampersand, whose value is a screen position.

HOME, TAB, BACKTAB and NEWLINE

Move the cursor to the first modifiable screen field, to the field that follows or precedes the current cursor location, or to the first modifiable screen position that follows the current cursor row.

literal-data,

Enclosed in single or double quotes, is typed.

&variable-data

Contained in a named REXX variable, coded with a leading ampersand, is typed.

ERASE

Erases the remainder of the modifiable screen field beginning at the current cursor position.

SELECT and DESELECT

Keywords select or deselect the pen detectable selection or attention field at the current cursor location. The operations simulate operations performed by a 3270 light pen or cursor select key, with the following exceptions:

- SELECT selects a pen selectable field even if the field is already selected,
- DESELECT deselects a pen selectable field even if the field is already deselected,
- If a pen attention field with a null designator is selected, a PEN keyword must be coded as the last TYPE command keystroke to send the pen aid and data.
- If a pen attention field with an ampersand designator (a simulated ENTER key) is selected, an ENTER keyword must be coded as the last TYPE command keystroke to send the data.

TOKEN

Identifies a CA Enterprise Tester password token, token, that follows the TOKEN keyword. token may be a literal value or a REXX variable coded with a leading ampersand, for example, &token, that contains a token. A token represents a password that is encrypted as described in Token Security. A token:

- is used to secure password data while Token Security is active;
- is converted to another form if Token Security is not active;
- may not be typed into a display field;
- is hidden in memory when it is typed;
- is not returned in PTEBUFF;
- can be decrypted only when used by it's owner;
- is decrypted only when it is sent to an online system; and
- replaces password data when a session is traced.

CODE

Identifies a CA Enterprise Tester password code, code, that follows the CODE keyword. code may be a literal value or a REXX variable, coded with a leading ampersand, for example, &code, that contains a code. A code is a password or other that is encoded as described in Code Security. A code:

- is used to secure encodable passwords while Code Security is active;
- is converted to another form if Code Security is not active;
- may not be typed into a display field;
- can be decoded correctly only when used by it's owner;
- is decoded when it is typed into a non-display field; and
- replaces encodable passwords when a session is traced;

USERMOD

Fetches data from fields of the virtual terminal display buffer in REXX variable, PTEBUFF, whose attribute values match a USERMOD value specified by the LOGON command that started a session, and rekeys the data into corresponding fields of the session's virtual terminal display buffer.

- A fully functional image of a virtual terminal screen may be displayed at a real terminal following any successful CA Enterprise Tester session command, by simply displaying an ISPF panel that defines a dynamic area named PTEBUFF and attributes defined by the CA Enterprise Tester LOGON command PANELDATA option.

- When a LOGON command used to start a CA Enterprise Tester session and an ISPF panel used to display the session specify the same USERMOD value, all data entered by a terminal user while a virtual terminal screen image is displayed may be rekeyed into the session's virtual terminal display buffer by issuing a single TYPE command that specifies USERMOD.
- A REXX exec illustrated in ISPF Session Panels on page 10-17, uses CA Enterprise Tester session commands and an ISPF panel to start and manage interactive sessions with other online systems while using any ISPF application.

ENTER, PF1-24, PA1-3, PEN, CLEAR and ATTN

Keywords send data to an online system that is formatted like the data sent when a like-named key is pressed on a real terminal keyboard. The AID keyword identifies a variable that follows, &aid, that contains the name of the key to be pressed to send data. Virtual terminal keys that send data may be coded as the last TYPE command keystroke only.

A TYPE command that does not specify a key that sends data to an online system completes when the last specified keyboard operation has been performed.

When a key that sends data is pressed, the virtual terminal keyboard is locked and data identical to the data sent by a real terminal is sent to the online system. After sending data, the TYPE command normally waits for and receives response messages from the online system, until a message is received that frees the terminal keyboard, before returning to the exec that issued the command.

The HANDLE command may be issued to instruct the TYPE command to return immediately after sending data or to return after receiving any message even if the message does not free the keyboard. A TYPE command completes with a TIMEOUT condition if a response is not received within a time limit specified by a previous WAIT command.

The TYPE command returns updated information in session variables, as documented in CA Enterprise Tester REXX Variables.

Conditions and Return Codes

At the completion of a TYPE command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. The following table describes TYPE command return codes and default REXX conditions that may be raised for some CA Enterprise Tester conditions.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
LOCKOUT	4		Times out waiting for port lock
TIMEOUT	8		Timed out waiting for response
LOGOFF	16		Session ended
ENDTRACE	20	HALT	Trace abended (ie., SD37)
CANCEL	24	HALT	CANCEL command processed
FAULT	34	HALT	Invalid command or system error

UNLOCK

The UNLOCK command unlocks the port being accessed by an exec when the command is issued. A port may be unlocked only by an exec that is running on the same port as the exec that issued the LOCK or ACCESS command that locked the port.

If another exec task was delayed because it attempted to use resources anchored to the port being unlocked, the exec task is resumed when the port is unlocked.

Example

```
" UNLOCK "
```

Format

```
UNLOCK
```

Conditions and Return Codes

At the completion of an UNLOCK command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. The following table describes UNLOCK command return codes and default REXX conditions that may be raised for some CA Enterprise Tester conditions.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
LOCKOUT	4		Timed out waiting for port lock
TIMEOUT	8		Time out waiting for response
LOGOFF	16		Session ended
ENDTRACE	20	HALT	Trace abended
CANCEL	24	HALT	CANCEL command processed
FAULT	64	HALT	Invalid command or system error

WAIT

The WAIT command waits for a specified period of time to elapse, or limits the amount of time that subsequent CA Enterprise Tester commands wait for online system responses, data to be queued, or locks.

Examples

```
" WAIT &mssth "

" WAIT  50000 on response "

" WAIT  forever on pull "
```

Format

```
WAIT hhmmssth | FOREVER

( ON event )
```

Descriptions

hhmmssth is a time interval, expressed in hours (hh), minutes (mm), seconds (ss) and tenths and hundredths of seconds (th), or the keyword, FOREVER. FOREVER is valid only when ON event is specified.

ON event is optional. If not specified, the WAIT command simply waits for the specified interval to elapse. ON indicates that the time limit applies only when waiting for a named event to complete.

- ON LOCK limits the amount of time that CA Enterprise Tester commands will wait for a resource lock, or the time that the LOCK command will wait for a port lock.
- ON PULL limits the amount of time that the CA Enterprise Tester PULL command will wait for data when a PULL command is issued while the accessed data queue is empty.
- ON RESPONSE limits the amount of time that a CA Enterprise Tester session command will wait for a required online system response to a virtual terminal input.

Conditions and Return Codes

At the completion of a WAIT command, the REXX variable, RC, indicates any CA Enterprise Tester condition encountered while executing the command. The following table describes WAIT command return codes and default REXX conditions that may be raised for some CA Enterprise Tester conditions.

Conditions	RC	Default Signal	Reason for Condition
	0		Command completed successfully
CANCEL	24	HALT	CANCEL command processed
FAULT	64	HALT	Invalid command or system error

Chapter 12: REXX Variables

This chapter documents special variables defined by CA Enterprise Tester to provide information to a REXX exec.

CA Enterprise Tester REXX Variables

CA Enterprise Tester commands return specific information to execs in CA Enterprise Tester defined REXX variables.

- All CA Enterprise Tester defined REXX variable names begin with the characters PTE. Execs that use CA Enterprise Tester commands for REXX should not create REXX variables that begin with PTE. Observing this convention ensures that there will be no conflict between user defined variables and current or future CA Enterprise Tester defined variables.
- PTEPORT, PTEAPORT, PTEPORTS, PTEINFO and PTEMESS are used to return unique information and are always returned using the defined simple variable names.
- All other CA Enterprise Tester defined variables are session variables, used by session commands to return information about a session on a CA Enterprise Tester port accessed by an exec. If an exec accesses sessions on more than one port, session information is not unique. Session information is returned to an exec using either simple or compound session variable names that are unique for each session accessed by an exec.
- All variable names defined below are simple variable names. Session commands return information using simple session variable names by default or when an exec accesses the session on the exec's port by issuing an ACCESS command that instructs session commands to return information using simple session variable names.
- Compound session variable names have a REXX stem that is a defined simple session variable name and a tail that is a session's port number. For example, PTEBUFF.2 is a compound session variable name that may be used by session commands to return the session information defined for PTEBUFF when an exec accesses the session on port 2. Session commands return information using compound session variable names when an exec accesses a session on any port, including the session on the exec's port, by issuing an ACCESS command that instructs session commands to return information using compound session variable names.

- Session commands return session variables if a session is active on the accessed port at the completion of the session command. Session commands drop all session variables except PTERSENS if a session is not active on the accessed port at the completion of the command. The value of a dropped variable symbol is the value of the symbol itself.

PTEALARM

When a message is received from an online system, PTEALARM is set to YES if the message sounded the terminal alarm, or is set to NO if the message did not sound the terminal alarm.

PTEAPORT

Every ACCESS command sets PTEAPORT to the number of the port being accessed by the exec. ACCESS EXECPORT or ACCESS * accesses the port on which an exec is executing. ACCESS NEWPORT or an ACCESS command that specifies no operands, assigns and accesses a previously unused port. ACCESS FREEPORT accesses a port that may have been previously used, but is not currently being used, or access a previously unused port. An ACCESS command may specify the number of the port to be accessed.

PTEASIZ

When a session is started, PTEASIZ is set to one of the values 24X80, 32X80, 43X80or 27X132, and indicates the alternate screen size established for the session.

PTEBUFF

PTEBUFF is a copy of the virtual terminal display buffer used for a CA Enterprise Tester session. The size of the buffer, in rows and columns, is indicated by the session variables, PTEROWS and PTECOLS. Screen rows are arranged serially in PTEBUFF, so that when PTECOLS is 80, screen row 2 begins at position 81 in PTEBUFF.

The format of the data and attributes stored in PTEBUFF may be specified by a QUERY SESSION command. In any other case, the format of PTEBUFF is determined by the BASICDATA, EXTDSDATA, PANELDATA, IMAGEDATA or ATTRIBUTES options specified by the LOGON command that started the session.

PTECC

A chain is a complete message sent from or to a terminal. PTECC is a count of the outbound chains received at a virtual terminal during the execution of the previous CA Enterprise Tester session command. PTECC is set to 0 (zero) if no message is received during the execution of an INVITE or DELAY command.

PTECOL

A virtual terminal cursor works like a real terminal cursor. An online system may send orders that move the cursor to any display position. The TYPE command moves the cursor while keying data into a virtual terminal display, when tab or other keys are used, and moves the cursor to any specified display position. At the completion of any CA Enterprise Tester session command, PTEROW and PTECOL identify the current cursor row and column in the virtual terminal display buffer.

PTECOLS

PTECOLS contains the current number of virtual terminal screen columns. The possible values of PTECOLS are fixed when default and alternate screen sizes are established during session initiation. The current value of PTECOLS at any point in time is controlled by the online system using 3270 ERASE-WRITE or ERASE-WRITE-ALTERNATE commands.

PTECSRP

During the execution of a TYPE command, the cursor is advanced as data is keyed into a virtual terminal screen. When an online system sends a terminal message, the message may reposition the cursor. Whenever the cursor is repositioned, PTECSRP is set to the position of the cursor within PTEBUFF. When the cursor is at row 1, column 1, PTECSRP is 1. When the cursor is at row 2, column 1 of an 80-column screen, PTECSRP is 81.

PTEDSIZ

When a session is started, PTEDSIZ is set to one of the values 24X80, 32X80, 43X80or 27X132, and indicates the default screen size established for the session.

PTEEAB

PTEEAB is an image of a virtual terminal Extended Attribute Buffer used to store 3270 extended attribute data that correspond to the field attributes and data stored in PTEBUFF. Refer to 3270 Attribute Values for a description of the data contained in an extended attribute buffer.

When session variables are returned by a QUERY SESSION command, PTEEAB is returned if EXTDSDATA is specified, and is not returned if BASICDATA, PANELDATA or IMAGEDATA are specified. In any other case, when session variables are returned by a session command, PTEEAB is returned if the LOGON command that started the session specified EXTDSDATA or if the LOGON command started an EXTDS session and did not specify BASICDATA, PANELDATA or IMAGEDATA.

PTEINFO

If the QUERY command determines that a named CA Enterprise Tester resource is active, the resource name is returned in PTEINFO. If the QUERY command determines that a named CA Enterprise Tester resource is not active, the concatenation of NO and the resource name is returned in PTEINFO. For example, if the command QUERY SESSIONS finds a session active on any CA Enterprise Tester port, SESSIONS is returned in PTEINFO, otherwise, NOSESSIONS is returned in PTEINFO.

PTEKB

PTEKB is the state of a virtual terminal keyboard and is one of the following values:

- BUSY means that data has been typed but not entered and that additional data may be typed.
- FREE means that data may be typed, but no data has been typed since the keyboard was freed.
- INHIBITED means that a session is timed out, outbound or in another state that does not permit data to be typed, but a TYPE ATTN command is permitted.
- LOCKED means that a session has timed out in a state that does not permit the use of the TYPE statement.

  Note: Keyboard state names that have lower alphabetic values are more free and names with higher values are locked tighter. Data may be typed when PTEKB is LE "FREE", but may not be typed when PTEKB is GE "INHIBITED".

PTELUDS

When a session is started, PTELUDS is set to one of the values BASICDS or EXTDS, and indicates the 3270 data stream level supported by the virtual terminal used for the session.

PTEMESS

An error or information message may be stored in PTEMESS during the execution of any CA Enterprise Tester command. Selected messages are sent to the terminal or to another message destination automatically if a MONITOR command has been issued that specified SIGNALS, CONDITIONS or the PTEMESS variable name.

PTEPLU

When a session is started with any Primary Logical Unit (PLU), the network name of the online system is stored in PTEPLU. While most sessions are started by the LOGON command, sessions may also be started while executing any session command if an online system passes the PLU end of a session to another PLU. All session commands store the name of the current PLU in PTEPLU if a session is passed to another PLU.

PTEPORT

All ACCESS commands store the number of the port on which an exec is executing in PTEPORT.

PTEPORTS

A QUERY command may specify one of the keywords, SESSIONS, TRACES, TASKS or QUEUES, to obtain a list of the ports on which the named CA Enterprise Tester resource exists. For example, an exec may issue a QUERY SESSIONS command to set PTEPORTS to a list of port numbers on which CA Enterprise Tester sessions are active. If sessions are active on ports 3, 4 and 7, the value 3 4 7is stored in PTEPORTS.

PTEROW

A virtual terminal cursor works like a real terminal cursor. An online system may send orders that move the cursor to any display position. The TYPE command moves the cursor while keying data into a virtual terminal display, when tab or other keys are used, and moves the cursor to any specified display position. At the completion of any CA Enterprise Tester session command, PTEROW and PTECOL identify the current cursor row and column in the virtual terminal display buffer.

PTEROWS

PTEROWS is the current number of virtual terminal screen rows. The possible values of PTEROWS are fixed when default and alternate screen sizes are established during session initiation. The current value of PTEROWS at any point in time is controlled by the online system using 3270 ERASE-WRITE and ERASE-WRITE-ALTERNATE commands.

PTERSENS

If VTAM is unable to start a session requested by a LOGON command, or if an active session fails, or if data sent to an online system is rejected during the execution of a CA Enterprise Tester session command, the eight character sense code received from VTAM or the online system is stored in PTERSENS. Sense codes are defined by SNA and indicate the cause of a session failure or other negative response. If a LOGON fails, or a session fails unexpectedly, an exec may analyze the contents of PTERSENS to determine the cause. For example, 08570002 means the PLU is not active and 087D0001 indicates a session services path error, possibly resulting from an attempt to start a session with an undefined PLU. Unlike other session variables, PTERSENS is not dropped at the end of a session. General information on sense codes, may be found in SNA Sense Codes.

PTERTIME

When a message is received from an online system, response time is calculated by subtracting the time of the previous terminal input from the time the message is received, and the calculated response time, expressed in hundredths of a second, is stored in PTERTIME.

PTESENSE

When CA Enterprise Tester sends a negative response to a message received from an online system, the four character SNA sense code sent with the negative response is stored in PTESENSE. The sense code indicates the reason that the online system request was rejected. If a negative response is not sent during the execution of a CA Enterprise Tester session command, the value '0000' is stored in PTESENSE. General information on sense codes, may be found in SNA Sense Codes.

PTESIZE

When a CA Enterprise Tester session is started or when a 3270 ERASE-WRITE command is received from an online system, the default screen size, calculated as the default screen size rows times the default screen size columns, is stored in PTESIZE. When a 3270 ERASE-WRITE-ALTERNATE command is received, the alternate screen size is calculated as the alternate screen size rows times the alternate screen size columns, is stored in PTESIZE.

PTESLU

When a session is started by a LOGON command, the name of the VTAM APPL used as a CA Enterprise Tester virtual terminal is stored in PTESLU. SLU is an acronym for Secondary Logical Unit. The name is either a name supplied by the exec, or is an assigned name if a name was not supplied by the exec.

PTESTATE

PTESTATE is a name that describes the state of a CA Enterprise Tester session. PTESTATE may be any state name described in CA Enterprise Tester Logical Unit States, except OFFLINE. When a session ends, instead of storing the value OFFLINE, PTESTATE is dropped.

PTETTIME

When a TYPE command is used to send terminal input to an online system, think time is calculated by subtracting the time at which the previous message was received from the online system from the time of the terminal input, and the calculated think time, expressed in hundredths of a second, is stored in PTETTIME.

Chapter 13: Access Method

This chapter describes a call interface that allows user-written programs written in other languages to exploit CA Enterprise Tester communications capabilities.

This chapter includes the following sections:

- CA Enterprise Tester Access Method
- Terminal Common Area
- The LOGON CAL
- The CONVERSE CALL
- The INVITE CALL
- The LOGOFF CALL
- The WAIT CALL
- Access Method Program Sample

CA Enterprise Tester Access Method

The CA Enterprise Tester Access Method makes it possible for batch programs written in any programming language using standard CALL conventions to gain access to online systems using a virtual terminal.

A program that uses CA Enterprise Tester then becomes a virtual terminal operator. This virtual operator can logon to online systems, key in online requests, examine online system responses, and logoff of the systems

A program can simulate these functions on a virtual terminal using a standard programming language CALL to the CA Enterprise Tester Access Method. In COBOL, an Access Method CALL is coded as follows:

```
CALL PTEAM USING PTCA-COMM-AREA.
```

Prior to each CALL, the program must place a request code in a field of the CA Enterprise Tester Common Area, PTCA-COMM-AREA.

A LOGON request simulates the terminal logon sequence entered by an operator on 3270-family CRT in order to obtain a session with a desired online system.

After logging on, the program may place data into the CA Enterprise Tester screen, which is contained in the Common Area. A CONVERSECALL request simulates pressing the ENTER key, and then waiting until the online system has responded and freed the keyboard.

At the completion of a CONVERSE CALL, the batch program can examine the updated CA Enterprise Tester screen. When all requested online transactions have executed via successive CONVERSE CALLs, the batch program may terminate the session with the online system using an Access Method LOGOFF CALL.

Subsequent sections of this chapter provide detailed descriptions of the CA Enterprise Tester Common Area and its use in each of the various Access Method CALL requests; LOGON, CONVERSE, INVITE, LOGOFF and WAIT.

Terminal Common Area

Requests and responses pass between a batch program and CA Enterprise Tester by way of a CA Enterprise Tester Common Area. The Common Area is the single parameter required on every CALL to the CA Enterprise Tester Access Method.

The Common Area is mapped into the working storage of a COBOL program as follows:

```
01 PTCA-COMM-AREA    COPY PTEAMCLC.
```

Use the following table to define the Common Area for other program languages. The use of the various Common Area fields is fully described in subsequent sections that document the use of each of the CA Enterprise Tester Access Method CALL requests.

COBOL Name	Bytes	Format	Field Description
PTCA-FUNC-CODE	8	char	Request code: LOGON; CONVERSE; INVITE; LOGOFF or WAIT
PTCA-APPL-NAME	8	char	Requested or actual online system name
PTCA-TERM-NAME	8	char	Requested or actual terminal name
PTCA-TERM-TYPE	8	char	MOD2; MOD5; etc.
PTCA-TRACE-DDN	8	char	DD name to be used for session Screen Trace
PTCA-TRACE-CTL	1	char	Y = trace screens; N = don’t trace

COBOL Name	Bytes	Format	Field Description
PTCA-TRACE-FMT	1	char	I = Image trace F = Format trace U = Uppercase trace
PTCA-TRACE-PRF	1	char	Internal use - do not alter
PTCA-TRACE-EJE	1	char	Y - force page eject
----------	3	----------	Reserved
PTCA-LOGON-FPR	1	char	Y = prompt console if LOGON fails
PTCA-THNK-TIME	8	char	THINK time -hhmmssth
PTCA-WAIT-TIME	8	char	WAIT interval - hhmmssth
PTCA-SESS-STAT	2	binary	Access Method return code; 0 = request completed normally 4 = normal request completion, online system awaiting reply; 8 = session has been lost
PTCA-REAS-LOST	80	binary	Error reason message
PTCA-CHAIN-CNT	2	binary	Number of messages received during last CONVERSE or INVITE
----------	15	----------	Reserved
PTCA-BUFF-ROWS	2	binary	Current screen size – rows
PTCA-BUFF-COLS	2	binary	Current screen size - columns
PTCA-CUR-ROW	2	binary	Cursor location-row
PTCA-CUR-COL	2	binary	Cursor location - column
PTCA-AID-KEY	1	code	AID key to be sent; PTEAMCON defines aid values
----------	15	----------	Reserved
PTCA-3270-BUFF	3564	char	Screen buffer; screen rows x columns

Note: The PTCA Common Area is used as the parameter on a CALL by a batch program to the CA Enterprise Tester Access Method to pass requests and responses.

The LOGON CALL

A CA Enterprise Tester Access Method LOGON request is used to obtain a CA Enterprise Tester session between a batch program and an online system. A LOGON call must successfully complete prior to executing CONVERSE or INVITE calls.

The following CA Enterprise Tester Common Area fields, referred to by their COBOL field names, are applicable to the LOGON request:

Field Name	Description
PTCA-FUNC-CODE	The request code LOGON instructs the Access Method to obtain a session with an online system
PTCA-TERM-NAME	Specifies the VTAM Application Id that represents a CA Enterprise Tester virtual terminal. If a name is not specified, then an installation-specified exit program will assign a suitable name.
PTCA-APPL-NAME	Specifies the VTAM Application Id of the online system with which a session is desired. Consult with a system programmer at your installation for the VTAM names of the online systems defined in your network.
PTCA-TRACE-DDN	The DDNAME to be used for the optional session screen trace data set. If the DDNAME is not in the JCL, a screen trace is not produced. The default is PTETRACE. If the default or specified DDNAME is not specified in the Job Step's JCL, then screen traces are not produced.
PTCA-TRACE-CTL	May be set to Y (Yes) or N (No) prior to any CONVERSE or INVITE call to control whether inbound or outbound screens produced during request processing are to be traced.
PTCA-TRACE-FMT	May be set to I (Image), F (Format), or U (Uppercase), or to specify the content of traced screens. Image screen traces are edited, translating attributes, dark data and nulls to blanks, and graphic escape characters to dashes, (-). In uppercase screen traces, all lower case alphabetic characters are converted to uppercase, allowing a trace to be printed on a printer that does not support lowercase characters.

Field Name	Description
PTCA-LOGON-FPR	If set to Y and the logon attempt fails, the console operator will automatically be prompted with information and options to retry or reject the LOGON request. If it is not Y, a failing LOGON request is simply identified to the batch program via a return code. If PTCA-LOGON-FPR is set to Y, and a problem is encountered during logon processing, and the operator replies REJECT, (implying that the condition causing logon failure can not be determined or corrected), then PTCA-MESS-NUMB will contain the message number PTE00903. If PTCA-LOGON-FPR is set to Y, and a problem is encountered during logon processing, and the operator replies DOWN, (implying that the named online system is not executing), then PTCA-MESS-NUMB will contain the message number PTE00904.
PTCA-THNK-TIME	A think time interval may be specified in hours, minutes, seconds, tenths and hundreds of a second (HHMMSSTH). If a non-zero value is specified, a wait will occur prior to sending each terminal input to an online system. The value may be changed at any time during program execution. Any new value will apply to subsequent terminal inputs.
PTCA-SESS-STAT	The return code from the Access Method; 0=a session has been obtained; 8=a session could not be obtained.
PTCA-REAS-LOST	An 80-character message describing the reason for a LOGON failure.
PTCA-BUFF-ROWS	On completion of a successful LOGON, this field contains the initial number of screen rows as determined by the online system. The size may vary during the session if alternate screen size transactions are invoked.
PTCA-BUFF-COLS	On completion of a successful LOGON, this field contains the initial number of screen columns as determined by the online system. The size may vary during the session if alternate screen size transactions are invoked.
PTCA-CUR-ROW	Contains the row at which the cursor is initially positioned.
PTCA-CUR-COL	Contains the column at which the cursor is initially positioned.

The following example illustrates a COBOL routine that issues the LOGON request:

```
LOGON-ROUTINE.
   MOVE 'LOGON' TO PTCA-FUNC-CODE.
   MOVE 'CICS1' TO PTCA-APPL-NAME.
   MOVE SPACES  TO PTCA-TERM-NAME.
   MOVE 'Y'     TO PTCA-TRACE-CTL
                   PTCA-LOGON-FPR.
   MOVE 'I'     TO PTCA-TRACE-FMT.
   CALL PTEAM USING PTCA-COMM-AREA.
   IF PTCA-SESS-STAT GREATER THAN ZERO
        GO TO LOGON-FAILED.
```

The CONVERSE CALL

A CA Enterprise Tester Access Method CONVERSE request is used to send a CA Enterprise Tester screen to an online system for processing. A LOGON call must have successfully completed prior to issuing a CONVERSE call.

The following CA Enterprise Tester Access Method Common Area fields, referred to by their COBOL field names, are applicable to the CONVERSE request:

COBOL Field Name	Description
PTCA-FUNC-CODE	The request code CONVERSE instructs the Access Method to send the current contents of the CA Enterprise Tester screen to the online system, and to wait until the response from the online system has been received.
PTCA-TRACE-CTL	Controls whether or not session screens resulting from the CONVERSE are traced. If a PTETRACE DD statement is present in the job step's JCL and PTCA-TRACE-CTL is Y, then Screen Trace records will be written that illustrate any inbound and outbound messages that result from CONVERSE processing.
PTCA-TRACE-FMT	Controls the format of traced screens when screen trace is active. If set to I, dark data, attributes and nulls contained in the traced screen will be translated to blanks, resulting in a traced screen that appears as a screen image. If set to F, screen data is written as is, resulting in traced screens that accurately represent screen format.

COBOL Field Name	Description
PTCA-SESS-STAT	The return code from the Access Method: 0 = the CONVERSE request completed normally; 4 = the CONVERSE request completed normally but the session remains in bracket and inbound until a terminal input is received; 8 = the session has been lost.
PTCA-REAS-LOST	An 80-character message describing the reason for the loss of a session.
PTCA-CHAIN-CNT	After each CONVERSE call, this field will contain a count of the number of messages received from the online system during processing of the request.
PTCA-BUFF-ROWS	At the completion of each CONVERSE call, this field contains the current number of screen rows as determined by the online system. The size may vary during successive CONVERSE calls if alternate screen size transactions are invoked. This field may be altered only by the online system through the use of the 3270 ERASE WRITEand ERASE WRITE ALTERNATE commands.
PTCA-BUFF-COLS	At the completion of each CONVERSE call, this field contains the current number of screen columns as determined by the online system. The size may vary during successive CONVERSE calls if alternate screen size transactions are invoked. This field may be altered only by the online system through the use of the 3270 ERASE WRITE and ERASE WRITE ALTERNATE commands.
PTCA-CUR-ROW	The batch program may position the cursor to any row prior to a CONVERSE call. At the completion of the call, the field contains the row to which the cursor was positioned by the online system.
PTCA-CUR-COL	The batch program may position the cursor to any column prior to a CONVERSE call. At the completion of the call, the field contains the column at which the cursor was positioned by the online system.
PTCA-AID-KEY	If an AID key other than ENTER is used, the batch program may specify which AID key, ENTER CLEAR, PA1-PA3or PF1-PF24, is to be sent to the online system. The COBOL copybook, PTEAMCON, includes named constant fields which contain the correct values for each AID key.
PTCA-3270-BUFF	3564 char. terminal screen - rows x columns. See CA Enterprise Tester Screen Format for more information.

CA Enterprise Tester Screen Format

The CA Enterprise Tester screen is a simple table of screen rows. Following the LOGON call and every CONVERSE call, PTCA-BUFF-ROWS contains the current number of screen rows as specified by the online system through the use of the 3270 ERASE WRITE or ERASE WRITE ALTERNATE commands and the SNA BIND upon which the session was established at LOGON time. PTCA-BUFF-COLS indicates the length of each row. The COBOL PTCA-3270-BUFF is redefined to provide screen definitions for 24x80, 32x80, 43x80, and 27x132 screen sizes. The user can redefine the screen as needed.

The CA Enterprise Tester screen is contained in host virtual memory. Since host virtual memory is unprotected, it is possible to overlay attributes and data contained in fields protected by 3270 protect attributes. A program that uses the CA Enterprise Tester Access Method is responsible for avoiding invalid screen modification that may produce unwanted or unpredictable results.

When a CONVERSE call executes, the Access Method sends a message to the connected online system. The message data constructed is identical to the data sent by a 3270-family control unit. For a request that specifies a CLEAR or PA1-PA3 aid key, the data stream consists of only the aid character.

When you specify other aid keys, data is sent in accordance with the 3270 read-modified operation. In addition to the aid key and cursor position, the address of and data contained in screen fields, which are controlled by attributes that have the modified data tag turned on, are sent to the online system.

PTEAMCON

A batch program that calls the CA Enterprise Tester Access Method is responsible to alter the attributes of the screen fields that it modifies, so that the modified data tag is turned on when a CONVERSE call is executed. The COBOL copybook, PTEAMCON, contains a named constant for each possible 3270 attribute. A batch program moves the new attribute whose modified data tag is on into the CA Enterprise Tester screen. This action overlays the attribute that controls the field that was modified by the program.

CONVERSE Example

The following example illustrates how a COBOL routine that issues the CONVERSE request may be coded:

```
CONVERSE-ROUTINE.
   MOVE 'CEMT I DA(USERFILE)' TO PTCA-LINE(1).
   MOVE PTCA-ENTER   TO PTCA-AID-KEY.
   MOVE 'CONVERSE'   TO PTCA-FUNC-CODE.
   CALL PTEAM USING PTCA-COMM-AREA.
   IF PTCA-SESS-STAT EQUAL 8
       GO TO CONVERSE-FAILED.
```

The INVITE CALL

A CA Enterprise Tester Access Method INVITE request is used to control the flow of transactions on a session when both the batch program and the online system initiate transactions on the session. INVITE checks for an online system request to initiate a transaction, and if a request is pending, permission to start the transaction is granted, and the message from the online system is received and placed in the CA Enterprise Tester screen buffer.

The following terminal Common Area fields, referred to by their COBOL field names, are applicable to the INVITE request:

Field	Description
PTCA-FUNC-CODE	The request code INVITE instructs the Access Method to grant permission to an online system to initiate a transaction, and if a request is pending, and to receive any messages sent by the online system as a result of the permission granted.
PTCA-SESS-STAT	The return code from the Access Method: 0=the INVITE request completed normally; 4=the INVITE request completed normally but a conversational online transaction is waiting for an input message; 8=the session has been lost.
PTCA-REAS-LOST	An 80-character message describing the reason for the loss of a session.
PTCA-CHAIN-CNT	Contains a count of the number of messages received from the online system during INVITE processing. A value of zero indicates that no online system message was waiting to be received at the time of the INVITE call.

If the online system initiates a transaction during INVITE processing, the ProTerm screen buffer, rows, columns and cursor position are updated to reflect the message received from the online system. Refer to The CONVERSE CALL for a description of these fields.

INVITE Example

The following example illustrates how a COBOL routine that issues the INVITE request may be coded:

```
INVITE-ROUTINE.
   MOVE 'INVITE' TO PTCA-FUNC-CODE.
   CALL PTEAM USING PTCA-COMM-AREA.
   IF PTCA-SESS-STAT EQUAL 8
       GO TO SESSION-FAILED.
   IF PTCA-CHAIN-CNT IS GREATER THAN ZERO
       GO TO ONLINE-SYS-STARTED-TRANSACTION.
```

The LOGOFF CALL

A CA Enterprise Tester Access Method LOGOFF CALL requests normal termination and cleanup of a CA Enterprise Tester session with an online system. It is a good programming practice to request a LOGOFF to terminate every CA Enterprise Tester session obtained via a LOGON request. Ending execution of a batch program without terminating an active session via a LOGOFF CALL will cause abnormal termination of a session. Some online systems may react to abnormal session termination by putting the virtual terminal out of service. Untidy session terminations may cause session start-up problems for subsequent programs that attempt to logon using the terminal.

The following CA Enterprise Tester Common Area fields, referred to by their COBOL field names, are applicable to the LOGOFF request:

Field Name	Description
PTCA-FUNC-CODE	The request code LOGOFF instructs the Access Method to terminate an active session with an online system.

LOGOFF Example

The following example illustrates how a COBOL routine that issues the LOGOFF request may be coded:

```
LOGOFF-ROUTINE.
   MOVE 'LOGOFF' TO PTCA-FUNC-CODE.
   CALL PTEAM USING PTCA-COMM-AREA.
```

The WAIT CALL

The WAIT request provides a means by which a program can delay program execution for a specified period. You can use the WAIT request to monitor applications or to simply pace or slow down the rate of online transaction requests.

The following CA Enterprise Tester Common Area fields, referred to by their COBOL field names, are applicable to the WAIT request:

Field Name	Description
PTCA-FUNC-CODE	The request code WAIT instructs the Access Method to delay program execution for the time interval contained in PTCA-WAIT-TIME.
PTCA-WAIT-TIME	Specify a time interval expressed in hours, minutes, seconds, tenths and hundredths of seconds. The format is HHMMSSTH.

WAIT Example

The following example illustrates how a COBOL routine that issues the WAIT request may be coded:

```
WAIT-ROUTINE.
   MOVE 'WAIT'      TO PTCA-FUNC-CODE.
   MOVE '00010000'  TO PTCA-WAIT-TIME.
   CALL PTEAM USING PTCA-COMM-AREA.
```

Access Method Program Sample

This section provides a sample COBOL program that utilizes the CA Enterprise Tester Access Method. All input to the sample program is via the JCL EXEC statement PARM field. The PARM field contains the name of an online system, the name of a CA Enterprise Tester virtual terminal, and a single transaction to be sent to the online system. The program checks all return codes and displays messages that indicate the results of program execution. The Access Method will automatically produce a screen image trace via the PTETRACE data set, of the screen sent to and received from the online system.

Sample JCL

The following example illustrates the JCL used to execute a sample program:

```
//jobname JOB . . .
//STEP010  EXEC PGM=PTEAMCLS,
//  PARM='PT01 CICS1   CSMT TERMNL,SIN,INQ,TERMID=PT01'
//PTETRACE  DD SYSOUT=T
//SYSOUT DD SYSOUT=T
```

Sample COBOL Program using the Access Method

The following example illustrates a COBOL program using the CA Enterprise Tester Access Method:

```
ID DIVISION.
PROGRAM-ID. PTEAMCLS.
REMARKS.
   THIS SAMPLE PROGRAM USES THE PROTERM TERMINAL
   ACCESS METHOD, TO LOGON TO AN ONLINE SYSTEM AND
   EXECUTE A SINGLE TRANSACTION, BASED ON INFO
   PROVIDED VIA THE JCL EXEC STATEMENT PARM FIELD.
   TO DOCUMENT WHAT HAPPENS, INCLUDE PTETRACE AND
   SYSOUT DD STATEMENTS.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
77  KEY-COUNT   PIC S9(4) COMP.
77  COL   PIC S9(4) COMP.
77  ROW   PIC S9(4) COMP.
77  PTEAM    PIC X(8) VALUE 'PTEAM'.
COPY PTEAMCLC.
COPY PTEAMCON
LINKAGE SECTION.
01  PARM-AREA.
    03  PARM-LEN   PIC S9(4) COMP.
    03  PARM-SLU   PIC X(8).
    03  PARM-PLU   PIC X(8).
    03  PARM-TRN   OCCURS 80 TIMES PIC X.
PROCEDURE DIVISION USING PARM-AREA.
    IF PARM-LEN LESS THAN 17
        DISPLAY 'INVALID PARM'
        GOBACK.
MOVE 'LOGON'     TO PTCA-FUNC-CODE.
MOVE PARM-SLU    TO PTCA-TERM-NAME.
MOVE PARM-PLU    TO PTCA-APPL-NAME.
CALL PTEAM USING PTCA-COMM-AREA.
IF PTCA-SESS-LOST
    DISPLAY 'LOGON FAILED'
    DISPLAY PTCA-REAS-LOST
    GOBACK.
MOVE 'WAIT'      TO PTCA-FUNC-CODE.
MOVE '00001000'  TO PTCA-WAIT-TIME.
CALL PTEAM USING PTCA-COMM-AREA.
DISPLAY 'WAITED 10 SECONDS'.
MOVE 'INVITE'    TO PTCA-FUNC-CODE.
CALL PTEAM USING PTCA-COMM-AREA.
IF PTCA-SESS-LOST
    DISPLAY 'INVITE FAILED'
    DISPLAY PTCA-REAS-LOST
    GOBACK.
```

```
MOVE 'CONVERSE'  TO PTCA-FUNC-CODE.
MOVE PTCA-CLEAR  TO PTCA-AID-KEY.
CALL PTEAM USING PTCA-COMM-AREA.
IF PTCA-SESS-LOST
    DISPLAY 'CLEAR FAILED'
    DISPLAY PTCA-REAS-LOST
    GOBACK.
COMPUTE KEY-COUNT EQUAL PARM-LEN -16.
PERFORM TYPE-TRAN VARYING COL FROM 1 BY 1
    UNTIL COL IS GREATER THAN KEY-COUNT.
MOVE 'CONVERSE'  TO PTCA-FUNC-CODE.
CALL PTEAM USING PTCA-COMM-AREA.
IF PTCA-TRAN-DONE
    GO TO DISPLAY-BUFF
   ELSE IF PTCA-CONT-TRAN
    DISPLAY 'CONVERSATION NOT SUPPORTED'
   ELSE
    DISPLAY 'SESSION LOST'
    DISPLAY PTCA-REAS-LOST.
   GO TO LOGOFF-CICS.
DISPLAY-BUFF.
  IF PTCA-MOD2PERFORM
    DISPLAY-MOD2 VARYING ROW FROM 1 BY 1
      UNTIL ROW IS GREATER THAN PTCA-BUFF-ROWS
    DISPLAY 'END OF MOD 2 BUFF'
   ELSE IF PTCA-MOD5
  PERFORM
    DISPLAY-MOD5 VARYING ROW FROM 1 BY 1
      UNTIL ROW IS GREATER THAN PTCA-BUFF-ROWS
    DISPLAY 'END OF MOD 5 BUFF'
   ELSE
    DISPLAY 'BUFFER NOT MOD 2 OR MOD 5'.
LOGOFF-CICS.
    MOVE 'LOGOFF' TO PTCA-FUNC-CODE.
    CALL PTEAM USING PTCA-COMM-AREA.
    GOBACK.
TYPE-TRAN.
    MOVE PARM-TRN(COL) TO PTCA-C(1, COL).
DISPLAY-MOD2.
    DISPLAY PTCA-LINE(ROW).
DISPLAY-MOD5.
    DISPLAY PTCA-MOD5-LINE(ROW).
```

Chapter 14: Logical Unit Codes

This chapter provides reference material that documents codes defined by CA Enterprise Tester and codes defined by SNA, VTAM and other systems that are used with CA Enterprise Tester facilities.

- 3270 Attribute Values
- CA Enterprise Tester Logical Unit States
- CA Enterprise Tester Session Status Flags
- SNA Request Header(RH) Codes
- SNA Request and Response Units(RU)
- VTAM Return Codes and Feedback

3270 Attribute Values

A SET statement that uses the ATTRIBUTES adjective may be used to put a list of the names of the 3270 attributes that describe a specified screen location in a specified variable field. The following table documents all possible attribute names.

Attribute Name	Description
BLINK	Is an extended data stream attribute that identifies a field attribute that begins a blinking screen field or a character screen location that has the blinking character attribute.
BLUE	Is an 3270 extended attribute that identifies a field attribute that begins a blue screen field or a character location that has the blue character attribute.
BRIGHT	Is a basic data stream attribute that identifies a field attribute that begins a high intensity screen field.
CHARACTER	Identifies a screen location that is a character.
DARK	Is a basic data stream attribute that identifies a field attribute that begins a nondisplay screen field.
FIELD	Identifies a screen location that is a field attribute.
GRAPHIC	Identifies a screen location that was identified using the 3270 graphic escape order or the Set Attribute character set X'F1'.

Attribute Name	Description
GREEN	Is an extended data stream attribute that identifies a field attribute that begins a green screen field or a character location that has the green character attribute.
MODIFIABLE	Identifies a character screen location that may be modified.
MODIFIED	Is a basic data stream attribute that identifies a field attribute whose modified data tag (MDT) is set. The MDT may be set by an online system or by keying data into the screen field.
NORMAL	Is a basic data stream attribute that identifies a field attribute that begins a normal intensity screen field.
NUMERIC	Is a basic data stream attribute that identifies a field attribute that begins a screen field that will accept numeric input only.
PINK	Is an extended data stream attribute that identifies a field attribute that begins a pink screen field or a character location that has the pink character attribute.
PROTECTED	Is a basic data stream attribute that identifies a field attribute that begins a protected screen field.
RED	Is an extended data stream attribute that identifies a field attribute that begins a red screen field or a character location that has the red character attribute.
REVERSE	Is an extended data stream attribute that identifies a field attribute that begins a reverse video screen field or a character screen location that has the reverse video character attribute.
SELECTABLE	Is a basic data stream attribute that identifies a field attribute that is selectable using a light pen or cursor select key.
SKIP	Is a basic data stream attribute that identifies a field attribute that begins a protected screen field that the cursor will skip over when data is keyed into the preceding screen location.
TURQUOISE	Is an extended data stream attribute that identifies a field attribute that begins a turquoise screen field or a character location that has the turquoise character attribute.
UNDERSCORE	Is an extended data stream attribute that identifies a field attribute that begins an underscored screen field or a character screen location that has the underscore character attribute.

Attribute Name	Description
UNMODIFIABLE	Identifies a character screen location that is in a protected screen field.
UNPROTECTED	Is a basic data stream attribute that identifies a field attribute that begins an unprotected screen field.
WHITE	Is an extended data stream attribute that identifies a field attribute that begins a white screen field or a character location that has the white character attribute.
YELLOW	Is an extended data stream attribute that identifies a field attribute that begins a yellow screen field or a character location that has the yellow character attribute.

It is recommended that CA Enterprise Tester scripts use the SET statement ATTRIBUTES adjective and the IF statement CONTAINS condition to examine attributes rather than to use the values documented below. The recommended method creates terminal type independent scripts and provides access to all attributes, not just basic field attributes.

CA Enterprise Tester virtual terminal buffers are in a CA Enterprise Tester screen format trace. The information presented here is intended as an aid in interpreting the meaning of attributes when viewed in a trace data set. Refer to The Terminal Screen for a general description of CA Enterprise Tester virtual terminal buffers.

The following table documents the value of basic field attributes when stored in either a basic or extended CA Enterprise Tester virtual terminal screen buffer.

Basic Bin.	Basic Hex.	Panel Hex.	EXTDS Char.	EXTDS Hex.	Basic Field Attributes
1	01	01		40	UNPROTECTED NORMAL
2	02	01	A	C1	UNPROTECTED NORMAL MODIFIED
3	03	01	D	C4	UNPROTECTED NORMAL SELECTABLE
4	04	01	E	C5	UNPROTECTED NORMAL SELECTABLE MODIFIED
5	05	05	H	C8	UNPROTECTED BRIGHT SELECTABLE
6	06	05	I	C9	UNPROTECTED BRIGHT SELECTABLE MODIFIED

Basic Bin.	Basic Hex.	Panel Hex.	EXTDS Char.	EXTDS Hex.	Basic Field Attributes
7	07	07	<	4C	UNPROTECTED DARK
8	08	07	(	4D	UNPROTECTED DARK MODIFIED
9	09	01	&	50	UNPROTECTED NORMAL NUMERIC
10	0A	01	J	D1	UNPROTECTED NORMAL NUMERIC MODIFIED
11	0B	01	M	D4	UNPROTECTED NORMAL NUMERIC SELECTABLE
12	0C	01	N	D5	UNPROTECTED NORMAL NUMERIC SELECTABLE MODIFIED
13	0D	05	Q	D8	UNPROTECTED BRIGHT NUMERIC SELECTABLE
14	0E	05	R	D9	UNPROTECTED BRIGHT NUMERIC SELECTABLE MODIFIED
15	0F	07	*	5C	UNPROTECTED NUMERIC DARK
16	10	07	)	5D	UNPROTECTED NUMERIC DARK MODIFIED
17	11	11	-	60	PROTECTED NORMAL
18	12	11	/	61	PROTECTED NORMAL MODIFIED
19	13	11	U	E4	PROTECTED NORMAL SELECTABLE
20	14	11	V	E5	PROTECTED NORMAL SELECTABLE MODIFIED
21	15	15	Y	E8	PROTECTED BRIGHT
22	16	15	Z	E9	PROTECTED BRIGHT MODIFIED
23	17	11	%	6C	PROTECTED DARK
24	18	11	_	6D	PROTECTED DARK MODIFIED
25	19	19	0	F0	SKIP NORMAL

Basic Bin.	Basic Hex.	Panel Hex.	EXTDS Char.	EXTDS Hex.	Basic Field Attributes
26	1A	19	1	F1	SKIP NORMAL MODIFIED
27	1B	19	4	F4	SKIP NORMAL SELECTABLE
28	1C	19	5	F5	SKIP NORMAL SELECTABLE MODIFIED
29	1D	1D	8	F8	SKIP BRIGHT SELECTABLE
30	1E	1D	9	F9	SKIP BRIGHT SELECTABLE MODIFIED
31	1F	19	@	7C	SKIP DARK
32	20	19	`	7D	SKIP DARK MODIFIED

The following table documents the meaning of the bits at each location in a CA Enterprise Tester virtual terminal extended attribute buffer.

Bit(s)	Value	Meaning
0		Distinguishes field attributes from characters
	0	Screen location is a character
	1	Screen location is a field attribute
1-3		field or character color - a 3-bit number
	0	default color
	1	blue
	2	red
	3	pink
	4	green
	5	turquoise
	6	yellow
	7	neutral - white
4	`	undefined
5		character set (character location only)
	0	Base color
	1	Graphic Escape or Set Attribute character set X'F1'
6-7		field or character highlighting

Bit(s)	Value	Meaning
	0	default highlighting
	1	blink
	2	reverse video
	3	underscore

CA Enterprise Tester Logical Unit States

A CA Enterprise Tester logical unit (virtual terminal) is a finite state machine. This means that a virtual terminal is always in one of several states, and that the processing of session requests and events depends on the current state of the terminal. The current state of a virtual terminal is provided to CA Enterprise Tester scripts in a variable field, PTESTATE.

When session request unit logging is requested, by coding the RULOG keyword in a JCL PARM field or by requesting RU logging at the CA Enterprise Tester Interactive System Log panel, logged messages show the state of a virtual terminal as each session event is processed. State codes may also be displayed via status fields on the Control Panel or the Attention Panel.

The following list names and describes each CA Enterprise Tester logical unit State:

State	Description
OFFLINE	A session is not active or session termination is currently in progress.
INTERACTIVE	A session is not active. However, if an online system has sent a BIND request, a session may be started.
SETLOGON	CA Enterprise Tester has issued a SETLOGON macro statement while preparing to start a session, and is waiting completion of the VTAM RPL based request.
REQSESS	CA Enterprise Tester has issued a REQSESS macro statement to request a session with a named online system, and is awaiting completion of the VTAM RPL based request.

State	Description
PENDBIND	CA Enterprise Tester is waiting for a BIND request from an online system. A BIND request is the SNA request unit that is sent from an online system to a terminal to start a session. The PENDBIND state may be entered following the successful completion of a REQSESS macro or after processing an UNBIND request that indicated that another BIND was forthcoming.
OPNSEC	CA Enterprise Tester has issued an OPNSEC macro statement to accept a session with an online system, and is awaiting completion of the VTAM RPL based request.
RESET	CA Enterprise Tester is waiting for an SNA Start Data Traffic (SDT) command. The RESET state is entered following the successful completion of an OPNSEC macro statement or after processing an SNA CLEAR request that was sent by an online system to stop session traffic.
SESSIONC	CA Enterprise Tester has issued a SESSIONC macro statement to reject an unacceptable BIND request, and is awaiting completion of the VTAM RPL based request.
CONTENTION	A session is currently active and is between SNA brackets. Either the online system or the virtual terminal may send a message on the session.
ONLINE	The ONLINE state is the same as the CONTENTION state, but additionally indicates that this is the first time that the session has entered the contention state since accepting a BIND request to start a session. This state value may appear only in the Script Language variable field, PTESTATE.
PENDBB	CA Enterprise Tester has received and positively responded to an SNA BID request sent by an online system, and is currently waiting on a data request that specifies begin-bracket.
INBOUND	The session is in the in-bracket state and the session direction is inbound. The online system is waiting on the next normal flow message on the session to be sent from the terminal to the online system.

State	Description
SEND	CA Enterprise Tester is currently sending a data chain to an online system. The request may be a normal flow message that is sent when an ENTER, PF, PA or CLEAR key is pressed, may be a response to a read buffer, read modified, or read partition query request sent by the online system, or may be a null data request containing the change direction indicator that is sent in response to a SIGNAL request that was received from the online system. CA Enterprise Tester is awaiting the completion of a VTAM RPL based SEND request.
OUTBOUND	The session is in the in-bracket state, is between chains, and CA Enterprise Tester is waiting for the next normal flow session message to be sent from the online system to the terminal. The virtual terminal keyboard is inhibited with the exception of the Attention key.
RECEIVE	CA Enterprise Tester is currently receiving a data chain from an online system, and is awaiting the completion of a VTAM RPL based RECEIVE request.
REJECT	CA Enterprise Tester has sent a negative response to an invalid request that was sent by the online system to the virtual terminal. The rejected request was not the last or only element in the chain. While in the REJECT state, CA Enterprise Tester is waiting for the last element of the rejected chain or for another host recovery action.
ERP1	CA Enterprise Tester has sent a negative response to an invalid request that was sent by the online system to the virtual terminal. The entire rejected chain has been received and discarded. While in the ERP1 state, CA Enterprise Tester is waiting for a valid host recovery action to return the session to a normal session state.
SHUTC	CA Enterprise Tester has sent a SHUTC (shutdown complete) request in response to a SHUTD (shutdown) request previously received from an online system. While in the SHUTC state, CA Enterprise Tester is waiting for the completion of the VTAM RPL based SEND request.

State	Description
SHUT	CA Enterprise Tester has previously received a SNA SHUTD request and has responded by sending a SHUTC request. The session is now in the shut down state, meaning that normal flow messages may no longer be sent on the session. CA Enterprise Tester is waiting for the online system to clear the session and restart data traffic or to terminate the session.
TERMSESS	In response to an application request, or after processing an irresolvable error, CA Enterprise Tester has issued a TERMSESS macro statement to terminate the session between a virtual terminal and an online system. CA Enterprise Tester is waiting for the completion of the VTAM RPL based request.

CA Enterprise Tester Session Status Flags

This section documents the meaning of session status flags that appear in some CA Enterprise Tester messages. Messages that display the current settings of the flags refer to this documentation.

Session status flags are displayed as a three-byte hexadecimal value. The meaning of each bit, when on, is as follows:

Byte.Bit	Description
1.0	A session is currently active.
1.1	Session data traffic is active.
1.2	The session is currently in bracket.
1.3	An RU chain is currently being sent or received.
1.4	The current session direction is inbound to the online system.
1.7	The virtual terminal is in an active state.
2.0	A session was started during the current application.
2.1	A BID or begin bracket request was rejected during the current application call.
2.2	An SNA CLEAR request was processed during the current application call.
2.4	A negative response was sent to a request that was processed during the current application call.

Byte.Bit	Description
2.5	The session is timed out.
2.6	A request was received during the application call that sounded the terminal alarm.
2.7	The current application call was an invalid request.
3.0	A SHUTD request was received; shutdown is in progress.
3.2	A rejected request chain is being flushed.
3.3	Go to ERP1 state after flushing a rejected request chain.
3.7	PTECS is currently executing.

SNA Request Header (RH) Codes

CA Enterprise Tester messages that document a request or response unit (RU) that is sent or received on a CA Enterprise Tester session, contain a string of codes that document the SNA Request Header (RH) that was carried with the RU. RH flags are set by the sender of a message to control bracket, direction and chain states. In general, RH flags change session states when accepted by the receiver of a message. The code, RH, identifies the beginning of a string of Request Header codes. Following RH is a string of one or more codes, separated by dashes, that document particular RH flags. For example:

```
RH-BB-OIC-CD
```

The following list documents the meaning of RH codes.

RH Code	Description
BB	Indicates a request to being an SNA bracket.
EB	Indicates a request to end an SNA bracket.
FIC	Indicates that the message is the first element in a multiple element chain.
OIC	Indicates that the message is the only element in a chain.
MIC	Indicates that the message is a middle element in a multiple element chain.
LIC	Indicates that the message is the last element in a multiple element chain.

RH Code	Description
CD	Indicates a request to change the direction of message flow on a session.
RR	Indicates a positive response to the request was requested. An RH that does not contain RR indicates that only an exception response was requested.

SNA Request and Response Units (RU)

CA Enterprise Tester messages that document a request or response unit (RU) that is sent or received on a CA Enterprise Tester session, contain the name of the RU. The following list names and describes each RU supported by CA Enterprise Tester. If CA Enterprise Tester receives an unknown RU, it describes the RU as RU-xxxxxx, where xxxxxx is the hexadecimal value of the three-byte request unit identification contained in VTAM RPL field, RPLCNTRL.

RU	Description
BID	SNA LU type 2 protocol, implemented by both CA Enterprise Tester virtual terminals and real 3270-compatible terminals, defines rules to resolve contention conditions that arise when both a terminal and an online system send a message to the other party at the same time. The rule specifies that a message sent from a terminal to an online system has priority over a message sent from an online system, and that if a contention condition arises, that the terminal message is to be processed by the online system and that the message sent by the online system is to be rejected by the terminal. A BID command may be sent by an online system to a terminal to request permission to send a message. If a BID command is received at a terminal while in the contention state, the terminal keyboard is locked, a positive response is sent to the online system, and the terminal waits for a message from the online system. If a terminal has just sent a message when a BID is received, a negative response is sent to the BID, and the terminal waits for the online system to process the message that was previously sent.

RU	Description
BIND	A BIND request is sent from an online system to a terminal to start a session. A BIND contains session parameters that represent rules to be observed by the online system and the terminal for the duration of the session being started, and contains information about the capabilities of the terminal, including screen sizes and the 3270 data stream level to be supported. When a session is requested at a terminal or at a virtual terminal, information, equivalent to the information contained in a BIND, is obtained from a logmode entry in a VTAM logon mode table that is associated with the terminal or virtual terminal that is requesting the session. This information is provided to an online system as part of a request for a session. The online system may use, modify, or replace this information when specifying the session parameters to be included in the BIND that is sent to the terminal.
CANCEL	A CANCEL RU may be sent to a terminal by an online system to cancel a partial chain of DATA request units that was previously sent to the terminal. This may be done when a DATA RU sent by an online system to a terminal is rejected by the terminal prior to sending the last element in a DATA RU chain.
CHASE	A CHASE RU may be sent to a terminal by an online system to verify that all proceeding messages sent to the terminal have been received and processed. When the terminal receives a CHASE RU, a requested positive response is returned to the online
CLEANUP	A CLEANUP request may be sent to a terminal by VTAM to end a session between the terminal and an online system. A CLEANUP is a possible result of entering a transaction that requests that a session be ended or may be sent if a link that connects a terminal and an online system is lost. In any case, a CLEANUPRU indicates that a session has already been terminated by VTAM, and instructs a terminal to clean up any session related resources.
CLEAR	A CLEAR RU can be sent to a terminal to stop the flow of messages on a session and to inhibit terminal input. A CLEAR request may be implemented by an online system as a part of session recovery, to synchronize session flow following a session error. After sending a CLEAR request to a terminal, the online system must send an SDT(Start Data Traffic) request to resume the normal exchange of session messages.

RU	Description
DATA	A DATA request is used by both terminals and online systems to send messages to the other session partner. All messages that are sent to an online system when a terminal user presses the ENTER key, a PA key, a PF key, or the CLEAR key are DATA requests. All messages sent by an online system that cause data to be displayed at a terminal are DATA requests. Data requests that are sent by an online system to a terminal contain 3270 commands that erase or display data, and 3270 orders that position data and attributes on a terminal screen. DATA requests and other 3270 commands, that are not seen by a terminal user, may be sent by an online system to a terminal to request that the terminal send a DATA request containing the contents of a terminal buffer or a reply that describes the capabilities of the device.
NOTIFY	When a program, that is acting as a secondary logical unit (virtual terminal), requests a session with an online system, VTAM responds positively to the session request when the request is delivered to the online system. If the online system subsequently rejects the session request or terminates prior to starting the session, VTAM sends a NOTIFY request to the program (virtual terminal) to indicate that the session will not be established.
SDT	When a session is being started, only session control requests may be exchanged on a session. After a session is started, VTAM or an online system must send an SDT (Start Data Traffic) request to a terminal to enable the flow of normal session DATA requests. The flow of DATA requests may subsequently be stopped by an online system by sending a CLEAR request to a terminal. If DATA traffic is stopped, another SDT must be sent to the terminal to resume the exchange of normal DATA requests.
SHUTD	A SHUTD (shutdown) request may be sent to a terminal by an online system to begin the termination of a session or to stop data traffic. When a SHUTD request is sent to a terminal, the terminal responds by sending a SHUTC (shutdown complete) request to the online system. If a bracket is active on a session, a terminal defers sending the SHUTC request until the bracket is ended by the online system. After the SHUTC request is sent, DATA requests may not be sent by either session partner. An online system may terminate the session or may send CLEAR and SDT requests to resume the normal exchange of DATA requests.

RU	Description
SIGNAL	A SIGNAL request is sent from a terminal to an online system when the Attention key is pressed by a terminal user. The Attention key remains operational, even while the remainder of the keyboard is inhibited while waiting on a response to a terminal input. An online system may act on the SIGNAL request in any desired way. A SIGNAL request may also be sent from an online system to a terminal, to request a change in session direction. If a SIGNAL request is received at a terminal while a bracket is active on the session and the session direction is inbound, but the user has not pressed the enter key or another key that causes a message to be sent, the terminal will inhibit subsequent terminal input and send a DATA request that contains only a change direction indicator to change the session direction to outbound.
UNBIND	An UNBIND request is sent to a terminal when an online system terminates a session. An online system may also pass an active session with a terminal to some other online system. When a session is passed, an UNBIND is sent to the terminal, that indicates that a BIND from another online system is forthcoming. After a CA Enterprise Tester session is ended as a result of receiving an UNBIND request, if the UNBIND indicates a BIND is forthcoming, CA Enterprise Tester waits for and processes the subsequent BIND in exactly the same manner as a BIND that resulted from an original session request.

VTAM Return Codes and Feedback

CA Enterprise Tester messages that document the completion of RPL based VTAM requests, contain VTAM return and feedback codes. VTAM return and feedback codes are identified in messages by the keyword RCFB, a dash, a one byte return code and a one byte feedback code. Return and feedback codes are expressed in hexadecimal. For example:

```
RCFB-1012
```

Some commonly encountered return and feedback codes are documented with appropriate CA Enterprise Tester messages. Official documentation of all possible VTAM return and feedback codes may be found in a current IBM VTAM Programming manual. If you have access to NETVIEW, the RCFB rc fb command may be used to access VTAM return code and feedback documentation.

SNA Sense Codes

Each partner in an SNA session is called a logical unit or LU. Online systems and terminals are examples of logical units. A logical unit communicates with another logical unit by sending and receiving requests and responses. Requests are sent by an LU to request a service of another LU. For example, an online system may send a DATA request to a terminal to display data contained in the request. While processing a request, the receiving LU may encounter an error in the data. If an LU detects an error while processing a request, it may inform the sender of the request by sending a response. When an LU sends a request, the request may specify that the receiving LU is to send a response to acknowledge that the request was received and processed correctly. Requested responses that simply inform the sender of a request that the request was processed without a problem are called positive responses. Responses that report a problem with a request are called negative responses.

SNA defines codes, called sense codes, that may be used in a negative response to provide a reason for the negative response. A sense code is a two-byte value and is normally expressed in hexadecimal. For example, the sense code, 0813, may be sent in a response to tell the sender of a request that it may not begin a bracket. For some sense codes, an additional two bytes of data are defined that provides additional information about the sense code. For example, for sense code 0813, the two additional bytes of data may provide the reason that a bracket may not be started. Sense data 08130001 means that the sender of a request may not begin a bracket because a bracket has already been started.

Sense code definitions may be found in a current SNA Reference Summary, published by IBM. Alternatively, if you have access to NETVIEW, you may use the SENSE command to access sense code documentation.

While the general meaning of sense codes is defined by SNA, the actual sense codes contained in a negative response are specified by the logical unit that sends a negative response. A logical unit must choose the sense code that most closely describes the problem. In the real world, some liberty is taken by logical units when sense data is specified, possibly to provide more precise information than is defined by SNA. For example, SNA defines sense code 1001 to mean RU data error and defines only two values that may be found in the additional two bytes of sense data. The IBM 3174 controller sends the offset of a detected error as sense data, and sets the high order bit of the offset bytes to indicate that the value is the offset of the error. Sense data 10018007, sent to an online system by a3174 controller, means that the data contained in a request, contained an error, and that the error may be found at offset 7 in the data. This information may be found in a 3174 Establishment Controller Functional Description. In general, if a logical unit provides a definition of sense data, then the definition is likely to be more precise than the general SNA definition.

Many CA Enterprise Tester messages document available sense data. In a CA Enterprise Tester message, the expression, SENSE-ssssuuuu, documents available sense data, where ssss is the sense code and uuuu may be two additional bytes of sense data. Sense data that is contained in a response that is sent by CA Enterprise Tester is defined in the message description that documents the negative response. Sense data that is contained in a response that is sent by an online system may be documented in a manual that documents the online system software, or the online system may rely on documentation provided in the IBM SNA Reference Summary.

Appendix A: Sample Scripts

This appendix contains listings of completed CA Enterprise Tester scripts that illustrate script coding and that provide an idea of some of the things that may be accomplished using CA Enterprise Tester scripts.

- Sample CA Enterprise Tester Scripts
- Batch Multi-Port Applications
- A Central Encoded Password Maintenance System

Appendix A. Sample CA Enterprise Tester Scripts

This section presents a number of sample scripts. The samples are provided to illustrate script coding and to show ideas that may be useful. Some scripts may require tailoring to work at your installation. For example, sign on logic would need to be tailored to your security system.

Closing CICS Data Sets from a Batch Job Step

This sample script, member PTECLOS1 in the distributed CA Enterprise Tester script library may be executed in batch jobs to close and unallocate a list of data sets that are being used by a CICS system. The script allows data sets to be taken offline automatically when a batch file maintenance job is executed that requires exclusive control of a file. The script logs onto a CICS system whose name is supplied via the JCL EXEC statement PARM field, allowing the script to be used to close data sets in any CICS system. The DD names that are to be closed are specified via a DD statement whose name is CLOSLIST. Different close lists can be created for different online applications. The close procedure does not use the data set names contained in the CLOSLIST data shown below, however, their presence allows the same data set list to be used by the open script that is presented following the close sample. The following JCL may be used to execute the close script:

```
    //CLOSE  EXEC PGM=PTEBS,PARM='CICS=CICS1'
    //PTEIN    DD DSN=PTE.SCRIPT(PTECLOS1),DISP=SHR
    //PTELOG   DD SYSOUT=A
    //CLOSLIST DD *
    DDNAME1  DATA.SET.NAME1
    DDNAME2  DATA.SET.NAME2
    DDNAMEN  DATA.SET.NAMEN
 /*
The PTECLOS1 Script
*-------------------------------------------------------------------------*
```

```
* This script logs onto a CICS system specified via the EXEC statement *
* PARM field and closes and unallocates a list of ddnames specified    *
* via a dataset specified on a CLOSLIST DD. Each CLOSLIST record       *
* contains a DD name in record position 1. (Like PTEOPEN1 OPENLIST)    *
* This script uses the CICS CEMT transaction.                          *
*----------------------------------------------------------------------*
   set &l to length &cics
   if &l eq 0
   then
      log 'PARM=CICS=cicsid not specified, abending'
      log 'Correct parm and restart step.'
      abend 1
   endif
   handle logonfail
   handle sessionfail at clos0900
   logon &cics 24x80 24x80 basicds
   invite 1000
   link 'PTECODE'      /* see PTECODE sample script */
   type &pteuser tab code &code enter
   type clear
   type 'cemt' enter
   open input CLOSLIST endfile clos0300
clos0100 read CLOSLIST into &record
   set &ddname to &record(,1,8)
clos0200 set &waits to 0
   do while &waits < 12
      type home 'set dataset(' &ddname ') clo dis' erase enter
      if <3 22 *> contains 'c'
      then
        log 'close for dd=' &ddname ' completed successfully'
        goto clos0100
      endif
      wait 500     /* wait for completion */
      set &waits up 1
   enddo
   do until &reply = 'abend'
      wto 'PTECLOS1 was unable to close a CICS data
      set:' set &wto to image <1 1 80>     /* log image of CICS  */
      wto &wto     /* request/response  */
      set &wto to image <3 1 80>     /* on system console  */
      wto &wto
      set &wto to image <4 1 80>
      wto &wto
      set &wto to image <23 1 80>
      wto &wto
      wtor 'reply "retry", "bypass" or "abend"'
      if &ptereply eq 'retry'
         go to clos0200
      if &ptereply eq 'bypass'
         go to clos0100
```

```
   enddo
   abend 2
clos0300 close CLOSLIST
   type pf3
   type clear
   handle sessionfail on
   type 'cesf logoff' enter
   end
clos0900 log 'session not started or lost, abending'
   abend 3
```

Opening CICS Data Sets from a Batch Job Step

This sample script, member PTEOPEN1 in the distributed CA Enterprise Tester script library may be executed in batch jobs to allocate and open a list of data sets for use a CICS system. The script allows data sets to be put online automatically when at the completion of a batch file maintenance job that required excluse use of the data sets. The script logs onto a CICS system whose name is supplied via the JCL EXEC statement PARM field, allowing the script to be used to open data sets in any CICS system. The DD and data set names that are to be opened are specified via a DD statement whose name is OPENLIST. Different open lists can be created for different online applications. The following JCL may be used to execute the open script:

```
   //OPEN   EXEC PGM=PTEBS,PARM='CICS=CICS1'
   //PTEIN    DD DSN=PTE.SCRIPT(PTEOPEN1),DISP=SHR
   //PTELOG   DD SYSOUT=A
   //OPENLIST DD *
   DDNAME1  DATA.SET.NAME1
   DDNAME2  DATA.SET.NAME2
   DDNAMEN  DATA.SET.NAMEN
   /*
   The PTEOPEN1 Script
*-------------------------------------------------------------------------*
* This script logs onto a CICS system specified via the EXEC statement *
* PARM field and allocates and opens a list of data sets specified via *
* a data set specified on an OPENLIST DD. Each OPENLIST record contains*
* a DD name in record position 1, and a data set name in position 10.  *
* This script uses the CICS CEMT transaction.                          *
*-------------------------------------------------------------------------*
   set &l to length &cics
   if &l eq 0
   then
      log 'PARM=CICS=cicsid not specified, abending'
      log 'Correct parm and restart step.'
      abend 1
   endif
   handle logonfail
   handle sessionfail at open0900
```

```
   logon &cics 24x80 24x80 basicds
   invite 1000
   link 'PTECODE'     /* see PTECODE sample script */
   type &pteuser tab code &code enter
   type clear
   type 'cemt' enter
   open input OPENLIST endfile open0300
open0100 read OPENLIST into &record
   set &ddname to &record(,1,8)
   set &dsname to &record(,10,44)
open0200 type home 'set dataset(' &ddname ')'
                       ' object(' &dsname ') old' erase enter
   if <3 64 *> contains 'normal'
   then
      type home 'set dataset(' &ddname ') ena' erase enter
      if <3 64 *> contains 'normal'
      then
         log 'Open for DD=' &ddname ' completed successfully'
         go to open0100
      endif
   endif
   do until &reply = 'abend'
      wto 'PTEOPEN1 was unable to enable a CICS data set:'
      set &wto to image <1 1 80>     /* log image of CICS  */
      wto &wto     /* request/response  */
      set &wto to image <3 1 80>     /* on system console  */
      wto &wto
      set &wto to image <4 1 80>
      wto &wto
      set &wto to image <23 1 80>
      wto &wto
      wtor 'reply "retry", "bypass" or "abend"'
      if &ptereply eq 'retry'
         go to open0200
      if &ptereply eq 'bypass'
         go to open0100
   enddo
   abend 2
open0300 close OPENLIST
   type pf3
   type clear
   handle sessionfail on
   type 'cesf logoff' enter
   end
open0900 log 'session not started or lost, abending'
   abend 3
```

An Installation/Verification Job Manager

This sample illustrates a script that may be used to perform regression testing of batch jobs. The script submits a preplanned batch job and monitors job execution. When the batch job completes, the script archives the job output and compares job step completion messages contained in the output to corresponding messages contained in a previously archived job output listing. Differences are reported. Comments contained in the script describe setup and use. The script is provided in member PTEIVP of the distributed CA Enterprise Tester script library.

```
*--------------------------------------------------------------------*
* PTEIVP - Installation Verification Program                         *
*--------------------------------------------------------------------*
* This script is designed to:                                       *
* *  submit a job from a specified PDS member                       *
* *  monitor job execution and process job output when complete     *
* *  store job output as a member of a job output library           *
* *  select step completion messages from an old job output listing *
* *  select step completion messages from the new job output listing*
* *  compare step completion messages and identify differences      *
*--------------------------------------------------------------------*
* To get started, allocate a "Job Output List Library". Make it a PDS *
* with RECFM FB, LRECL 132 and BLKSIZE 27984.                       *
*                                                                   *
* Submit a planned job stream.  When it ends, store the job output  *
* listing in the Job Output List Library by entering the following  *
* TSO command: OUT jobname(job0nnnn) PRINT('list.lib.dsn(oldmem)')  *
*                                                                   *
* Execute this script to repeat the job.  The script submits the job, *
* monitors for job completion, stores the new job output listing as *
* a specified member of the Job Output List Library, then compares  *
* the old listing to the new listing and reports differences in     *
* job step completion messages.                                     *
*                                                                   *
* After starting a ProTerm session to TSO, and navigating to some   *
* ISPF panel, this script may be played on the port, and will use   *
* the TSO session to perform the functions described above.         *
*                                                                   *
* A script that logs on to TSO, signs on, and navigates to some ISPF *
* panel may then LINK to this script.   The "logon" script may be   *
* played interactively or be submitted as a batch job, as follows:  *
*                                                                   *
* Interactive: Play the "logon" script or start a ProTerm TSO session *
* and play this script on the same port.  At the Play specification *
* Panel, enter the parms described below.  When it ends, you may view *
* logged differences by scrolling at the monitor panel.  (it could  *
* write the messages to some data set if you wish.)                 *
*                                                                   *
* Batch: Submit the following JCL to execute your logon script that *
```

```
* links to this script.                                                *
*                                                                      *
* //stepname EXEC PGM=PTEBS,                                          *
* // PARM='JOB=dsn(mem),LIB=dsn,OLD=mem,NEW=mem,T=15'                  *
* //PTEIN  DD   DSN=script.lib.name(script-name),DISP=SHR             *
* //PTELIB  DD   DSN=script.lib.containing.this,DISP=SHR              *
* //PTELOG  DD   SYSOUT=*                                             *
*                                                                      *
* Job step completion mismatch messages are logged to PTELOG.         *
*                                                                      *
*----------------------------------------------------------------------*
* PARMs are: ** JOB=  dsn(mem) containing job to be analyzed.         *
* LIB=  Job Output List Library name.                                  *
* OLD=  old job list member name in LIB.                               *
* NEW=  new job list member name in LIB.                               *
* T=  maximum minutes to wait for submitted job to run.                *
*----------------------------------------------------------------------*
  set &tempold to &pteuser '.temp.' &OLD /* create some work         */
  set &tempnew to &pteuser '.temp.' &NEW /* dataset names            */
*----------------------------------------------------------------------*
* go to ISPF TSO COMMAND PROCESSOR and SUBMIT job stream, then        *
* verify submit success and save "jobname(jobid)"                      *
*----------------------------------------------------------------------*
  type home '=6' erase enter
  type "submit '" &job "'" enter
  set &row to &pterow
  set &row down 1
  if <&pterow 2 *> = '***'
and &row > 0
and <&row 2 *> = 'JOB'
 then
set &col to 6
do while <&row &col 1> > ' '
    set &col up 1
enddo
set &job-len to &col
set &job-len down 6
set &col up 1
if <&row &col *> ne 'SUBMITTED'
    go to submit-failed
 else
go to submit-failed
  endif
  set &job-id to <&row 6 &job-len>
*----------------------------------------------------------------------*
* Use STATUS command to monitor for job completion.                    *
* Issue a message and end if time limit expires.                       *
*----------------------------------------------------------------------*
  set &col to 7
  set &col up &job-len
```

```
  set &count to 0
  set &t from zoned &t
  do while &count < &T
set &count up 1
wait 10000
type enter
type home 'status ' &job-id erase enter
set &row to &pterow
set &row down 1
if <&pterow 2 *> = '***'
    and &row > 0
    and <&row 2 *> = 'JOB'
    and <&row 6 &job-len> = &job-id
    and <&row &col *> = 'ON OUTPUT QUEUE'
then
    go to job-ended
endif
  enddo
  go to timed-out-monitoring-status
*-------------------------------------------------------------------------*
* Use TSO OUT command to store new job list in job list library.      *
*-------------------------------------------------------------------------*
job-ended
  type enter
  type home "out " &job-id
" print('" &LIB "(" &NEW ")')"
erase enter
*-------------------------------------------------------------------------*
* Perform subroutine to create temporary edited copies of job lists   *
*-------------------------------------------------------------------------*
set &libmem to &OLD
set &tempdsn to &tempold  perform job-list-copy-and-edit
set &libmem to &NEW
set &tempdsn to &tempnew  perform job-list-copy-and-edit
*-------------------------------------------------------------------------*
* Perform subroutine to compare job step completion messages          *
*-------------------------------------------------------------------------*
  perform compare-step-completions
  log 'if played online, scroll up/down to view mismatches'
  pause end
*-------------------------------------------------------------------------*
* Message routines to deal with script problems                       *
*-------------------------------------------------------------------------*
timed-out-monitoring-status
  log 'time=' &T ' expired while waiting for job to end'
  log 'status at timeout was:'
  dump screen
  pause
  end
submit-failed
```

```
  log 'response to submit was not recognized'
  log 'submit response was:'
  dump screen
  pause
  end
*-------------------------------------------------------------------------*
* copy jcl list to temp dsn and delete all but step complete messages *
*-------------------------------------------------------------------------*
job-list-copy-and-edit
   type "alloc f(templist) da('" &tempdsn "') new "
      "dsorg(ps) recfm(f b) lrecl(132) blksize(27984) "
      "space(5,15) tracks" erase enter
   type home 'free f(templist)' erase enter
   allocate indd  dataset &LIB member &libmem shr
   allocate outdd dataset &tempdsn shr
   open input indd endfile job-list-edit
   open output outdd
job-list-copy
   read indd into &record
   write outdd from &record
   go to job-list-copy
job-list-edit
   close indd close outdd free indd free outdd
   type home '=2'       erase enter
   type tab  "'" &tempdsn "'" erase enter
   type home 'x all'    erase enter
   type home 'f all IEF142I'  erase enter  /* completed  */
   type home 'f all IEF472I'  erase enter  /* abended    */
   type home 'f all IEF272I'  erase enter  /* not executed */
   type home 'delete all x'   erase enter
   type home '=6'       erase enter exit
*-------------------------------------------------------------------------*
* compare step completion messages and document mismatches.           *
*-------------------------------------------------------------------------*
compare-step-completions
   log ''
   set &mismatches to 0
   allocate olddd dataset &tempold shr
   allocate newdd dataset &tempnew shr
   open input olddd endfile list-added-steps
   open input newdd endfile list-deleted-steps
compare-step-completion
  read olddd into &oldrec endfile list-added-steps
  read newdd into &newrec endfile list-deleted-steps
  if &oldrec ne &newrec then
set &mismatches up 1log 'STEP completion
mismatch ' &mismatches ':'log ' old:'
&oldrec(,1,130)log ' new:' &newrec(,1,130)
  endif
  go to compare-step-completion
```

```
list-added-steps  read newdd into &newrec endfile end-of-files
  log 'The following new step(s) not found in old job list'
list-added-step  set &mismatches up 1
  log ' new:' &newrec(,1,130)
  read newdd into &newrec endfile end-of-files
  go to list-added-step
list-deleted-steps
  log 'The following old step(s) not found in new job list'
list-deleted-step
  set &mismatches up 1
  log ' old:' &oldrec(,1,130)
  read olddd into &oldrec endfile end-of-files
  go to list-deleted-step
end-of-files  close olddd
  close newdd free olddd free newdd
  type home "delete '" &tempold "'" erase enter
  type enter
  type home "delete '" &tempnew "'" erase enter
  type enter
  log &mismatches ' step completion mismatches found'
  exit
```

Identifying Systems with Common Characteristics

Some scripts provide one or more functions which are available in a number of online systems. These online systems typically can be combined into logical groups based on criteria such the physical location at which the system is executing, whether it is considered to be a test or production system, the connectivity of the systems into networks or possibly something as simple as part of the system name.

Rather than hard code the system names into the script a technique to determine the names at execution time is desirable. For example a script to shut down a system could be executed each night for all test regions. The same script might run for all production regions at the end of the week. At another time it might be necessary to shut down all the systems at a particular site, regardless of whether the system is considered test or production. Another script might need to issue a CEMT command in a network of CICS systems.

This script, member PTESYS in the distributed CA Enterprise Tester script library, provides such a technique. By valuing the applicable variables and linking to PTESYS, the desired system names will be returned.

```
   The PTESYS Script
*-------------------------------------------------------------------------*
* SCRIPT NAME   => PTESYS    ** Created by    => Reed Software, Inc.  *
* Description   => This script returns the names of systems which     *
*    have one or more attributes in common.  Other                    *
*    scripts may LINK to this script with the                         *
*    appropriate values and be returned the system                    *
*    names which meet the criteria.                                   *
*                                                                     *
* Input Parms:                                                        *
* *
*     REQUEST: *
*  THE ITEM WHICH THE SYSTEMS ARE TO HAVE IN COMMON. THE  *
*  CURRENT VALID VALUES ARE:     *
*    1) NAME  - RETURNS ALL SYSTEM NAMES WHICH MATCH ON   *
*  CRITERIA.  A GENERIC NAME MAY BE SPECIFIED *
*  BY PLACING AN * (ASTERISK) AFTER THE LAST *
*  SIGNIFICANT CHARACTER. EXAMPLE:  PRODA*   *
*    2) PTAI  - RETURN SYSTEMS WHICH ARE PRODUCTION OR TEST *
*    3) NETW  - RETURN SYSTEMS IN A SPECIFIC NETWORK *
*    4) SITE  - RETURN SYSTEMS AT A SPECIFIC SITE   *
*     CRITERIA:      *
*  THE VALUE OF THE REQUEST THE SYSTEMS ARE TO HAVE IN    *
*  COMMON.     *
* *
* *
* Return Parms: *
* *
*     RESULT: *
*  THE NAMES OF THE SYSTEMS WHICH MEET THE SELECTION *
*   CRITERIA SUFFIXED WITH A PERIOD '.' *
*     RC:     *
*  RETURN CODE FROM THIS SCRIPT  *
*    0 -- SCRIPT COMPLETED NORMALLY    *
*      &RESULT CONTAINS THE OUTPUT FROM THE SCRIPT  *
*    8 -- AN UNEXPECTED CONDITION WAS ENCOUNTERED   *
*     MESSAGE: *
*  CONTAINS SUPPORTING TEXT AS TO WHY A NON-ZERO &RC IS   *
*    RETURNED TO THE LINKING SCRIPT.   *
* *
* *
* Files:  *
* *
* *
* Example script logic: *
* *
```

```
*   set &request  to 'NETW'    /* return system names  */ *
*   set &criteria to 'N1'  /*  in network 'N1' */ **   LINK 'PTESYS'        /*
LINK TO THE SCRIPT  */ *
*   IF &rc eq 0      *
*     then    *
* set &i to 1  *
* do forever   *
*   if &result(,&i,*) eq '.'  /* if the end of names */ *
*     then     /*   then get out     */ *
* leave *
*     else     *
* set &system to &result(,&i,8) /* current name  */ *
* . *
* . *
*   endif      *
*   set &i up 8    /* length of each system name  */ *
* enddo *
*     else    *
* log 'Return code from script "PTESYS"--' &rc      *
* log 'Message from script "PTESYS"--' &msg  *
*   endif     *
* *
*-------------------------------------------------------------------------*
* *
* EACH ENTRY CONSISTS OF THE FOLLOWING "FIELDS".  EVERYTHING PAST THE *
* SYS NAME IS USED TO MATCH AGAINST THE CRITERIA FIELD.      *
* *
* 8 CHARACTER SYSTEM NAME *
* 1 CHARACTER TO DENOTE EITHER PRODUCTION OR TEST REGION      *
* 2 CHARACTER CODE TO DENOTE A NETWORK -- IE CONNECTED SYSTEMS *
* 2 CHARACTER CODE TO DENOTE THE PHYSICAL SITE OF THE SYSTEM  *
* *
*-------------------------------------------------------------------------*
   SET &RC TO 0
   DROP &RESULT
   SET &SYSTEM-CRITERIA TO
      'PRODA1  PN1EC'
      'PRODA2  PN1EC'
      'PRODA3  PN1MW'
      'PRODA4  PN1MW'
      'PRODA5  PN1WC'
      'PRODA6  PN1WC'
      'PRODA7  PN1WC'
      'PRODB1  PN1WC'
      'PRODB2  PN1WC'
      'PRODC1  PN2MW'
      'PRODC2  PN2EC'
      'TESTA1  TN3EC'
      'TESTA2  TN3EC'
      'TESTA3  TN3MW'
```

```
   'TESTA4  TN3MW'
   'TESTA5  TN3WC'
   'TESTA6  TN3WC'
   'TESTB1  TN3WC'
   'TESTB2  TN3WC'
   'TESTB3  TN4MW'
   'TESTB4  TN4MW'
   'TESTC1  TN5MW'
   'TESTC2  TN5MW'
   '.'       /* END OF NAMES/CRITERIA    */
SET &CRIT-LEN TO LENGTH &SYSTEM-CRITERIA
IF &SYSTEM-CRITERIA(,&CRIT-LEN,*) NE '.' /* STOP LOOP!!!   */
THEN
   LOG '&SYSTEM-CRITERIA IN PTESYS MUST END WITH "."'
   ABEND 8
ENDIF
SET &CRIT-LEN TO LENGTH &CRITERIA
DO FOREVER
   IF &REQUEST EQ 'NAME'        /* REQUEST NAME OF SYSTEMS  */
   THEN
      SET &OFFSET TO 0
      IF &CRIT-LEN GT 0
      THEN
         IF &CRITERIA CONTAINS '*' /* GENERIC NAMES REQUEST */
         THEN
            COMPUTE &CRIT-LEN = &PTEARGI - 1
         ENDIF
   ENDIF
   PERFORM PTESYS-EXTRACT-SYSTEMS
   LEAVE
ENDIF
IF &REQUEST EQ 'PTAI'        /* CRITERIA IS PRODUCTION/TEST*/
THEN
   IF &CRIT-LEN NE 1
   THEN
      SET &RC TO 12   /* ERROR IN CRITERIA FIELD */
      LEAVE
   ENDIF
   SET &OFFSET  TO 8
   PERFORM PTESYS-EXTRACT-SYSTEMS
   LEAVE
ENDIF
IF &REQUEST EQ 'NETW'        /* CRITERIA IS NETWORK ID   */
THEN
   IF &CRIT-LEN NE 2
   THEN
      SET &RC TO 12   /* ERROR IN CRITERIA FIELD */
      LEAVE
   ENDIF
   SET &OFFSET  TO 9
```

```
      PERFORM PTESYS-EXTRACT-SYSTEMS
      LEAVE
   ENDIF
   IF &REQUEST EQ 'SITE'        /* CRITERIA IS SITE LOCATION */
   THEN
      IF &CRIT-LEN NE 2
   THEN
      SET &RC TO 12   /* ERROR IN CRITERIA FIELD  */
      LEAVE
   ENDIF
   SET &OFFSET  TO 11
   PERFORM PTESYS-EXTRACT-SYSTEMS
   LEAVE
ENDIF
SET &RC TO 8     /* UNKNOWN REQUEST    */
LEAVE
ENDDO
DO    /* ERROR MESSAGE BASED ON RC */
   IF &RC EQ 0
   THEN
      LEAVE
   ENDIF
   SET &RESULT TO '.'
   IF &RC EQ 8
   THEN
      SET &MESSAGE TO 'INVALID REQUEST - ' &REQUEST' - IN PTESYS'
      LEAVE
   ENDIF
   IF &RC EQ 12
   THEN
      SET &MESSAGE TO 'INVALID CRITERIA - ' &CRITERIA ' - FOR REQUEST ' &REQUEST
          IN PTESYS'
   LEAVE
   ENDIF
   SET &MESSAGE TO 'UNKNOWN RC - ' &RC ' IN PTESYS'
   ENDDO
   EXIT          /* RETURN TO LINKING SCRIPT     */
PTESYS-EXTRACT-SYSTEMS
   SET &LOC TO 1
   DROP &RESULT
   DO WHILE &SYSTEM-CRITERIA(,&LOC,*) NE '.'
      COMPUTE &CRIT-LOC = &LOC + &OFFSET
      DO IF &CRIT-LEN GT 0 THEN
      IF &SYSTEM-CRITERIA(,&CRIT-LOC,&CRIT-LEN) NE &CRITERIA(,1,&CRIT-LEN)
      THEN
   LEAVE
      ENDIF
ENDIF
   SET &RESULT TO &RESULT &SYSTEM-CRITERIA(,&LOC,8)
      ENDDO
```

```
   SET &LOC UP 13
ENDDO
SET &RESULT TO &RESULT '.'
EXIT
```

Creating Multiple Files From a Single File

Many times when a script is played on multiple ports to provide a simulated online production workload, a file can be used to extract the data which will be keyed into the application. In most instances it would be desirable to have different data being keyed on each port. This script, member PTESPLIT in the distributed CA Enterprise Tester script library, takes data from an input file and distributes the data to one or more existing datasets in a generation data group. By allocating a unique generation to each script being played in a multiple session test, different data may be keyed into the application from each port. This closely simulates a production environment in which multiple terminal operators enter the same transactions but with unique data.

The script uses many keyword parameters which allows the script to be quite flexible in determining such things as the number of datasets to be created, the number of records to be written to each output file and the order in which the output files will have the selected records written to them. In addition, an optional user script may be specified to allow for the output record to be formatted from the selected input record. These parameters as well as other are documented at the beginning of the script.

```
   The PTESPLIT Script
*-------------------------------------------------------------------*
* PURPOSE:  *
* *
* SPLIT RECORDS FROM A SINGLE INPUT FILE INTO ONE OR MORE EXISTING *
* GENERATION DATASETS. *
* *
* PARMS:    *
* *
* GDGBASE    - (REQUIRED) THE BASE NAME OF A GENERATION DATA *
*   GROUP (GDG) TO BE OPENED FOR OUTPUT.     *
* *
* GENS       - (REQUIRED) THE NUMBER OF EXITSING DATASETS IN THE   *
*   GDG TO WHICH SELECTED RECORDS WILL BE    *
*   WRITTEN.  *
* *
* CNT  - (REQUIRED) THE NUMBER OF RECORDS TO BE WRITTEN TO   *
*   EACH OUTPUT FILE.     *
* *
* ENDFILE    - (OPTIONAL) ACTION TO BE TAKEN IF THERE ARE NOT ENOUGH *
*   RECORDS ON THE INPUT FILE TO SATISFY THE *
*   REQUESTED NUMBER OF RECORDS TO BE WRITTEN *
*   TO THE OUTPUT FILES AS SPECIFIED BY THE  *
```

```
*   CNT PARM. *
*      "END"     THE DEFAULT, SPECIFIES THE SCRIPT WILL   *
*   COMPLETE AFTER ALL RECORDS FROM THE INPUT *
*   FILE HAVE BEEN PROCESSED.   *
*      "REUSE"   SPECIFIES THE RECORDS FROM THE INPUT FILE *
*   BE REUSED HOW EVER MANY TIMES IS NECESSARY *
*   IN ORDER TO OBTAIN THE DESIRED NUMBER OF *
*   RECORDS ON THE OUTPUT FILES. *
* *
* *
* FREQ       - (OPTIONAL) AN INTEGER VALUE DENOTING THE FREQUENCY  *
*   AT WHICH A RECORD FROM THE INPUT FILE    *
*   WILL BE SELECTED TO FOR OUTPUT PROCESSING. *
* *
*   THE DEFAULT VALUE IS 1, INDICATING EVERY *
*    RECORD WILL BE SELECTED.   *
* *
* ORDER      - (OPTIONAL) USED TO SPECIFY HOW RECORDS ARE TO BE    *
*   DISTRIBUTED TO THE OUTPUT FILES.   *
* *
*      "SEQ"     "CNT" RECORDS ARE WRITTEN TO THE FIRST   *
*   FILE, THEN "CNT" RECORDS TO THE SECOND FILE*
*   AND SO ON UNTIL "GENS" FILES HAVE BEEN   *
*   PROCESSED. **      "RAN"     RECORDS ARE WRITTEN TO ALL FILES AT THE  *
*   RATE.  FOR EXAMPLE, AFTER A RECORD IS    *
*   WRITTEN TO THE FIRST FILE, THE SECOND FILE *
*   WILL RECEIVE A RECORD, THEN THE THIRD FILE *
*   ETC UNTIL ALL FILES HAVE RECEIVED A RECORD *
*   AT WHICH TIME THE FIRST FILE WILL GET THE *
*   SECOND SECOND RECORD. *
* *
*      "SEQ" IS THE DEFAULT.     *
* *
* FORMAT     - (OPTIONAL) A SCRIPT TO BE EXECUTED JUST PRIOR TO THE *
*   SELECTED RECORD BEING WRITTEN TO THE     *
*   OUTPUT FILE.  THIS IS ACCOMPLISHED VIA   *
*   THE LINK VERB.  IF THE SPECIFIED SCRIPT  *
*   WISHES TO FORMAT THE OUTPUT RECORD, IT   *
*   SHOULD FORMAT THE OUTPUT RECORD IN ITS   *
*   ENTIRITY IN THE VARIABLE &OUTPUT-RECORD. *
*   THE SELECTED INPUT RECORD IS IN THE *
*   VARIABLE &INPUT-RECORD.  AFTER THE RECORD *
*   IS WRITTEN TO THE OUTPUT FILE      *
*   &OUTPUT-RECORD IS DROPPED.  *
* *
*  THE DEFAULT IS NO MEMBER, THEREFORE THE OUTPUT   *
*    RECORD WILL BE IDENTICAL TO THE INPUT RECORD.  *
* *
* *
* FILES:  *
```

```
* SYSUT1   -- INPUT FILE TO OBTAIN RECORDS.  *
* *
*-------------------------------------------------------------------*
   TIME
   LOG 'SCRIPT STARTED BY ' &PTEUSER
      ' AT ' &PTEHR ':' &PTEMIN ':' &PTESEC
   SET &SEL TO 0
PARM000 IF &CNT EQ ''
   THEN
      WTOR 'ENTER NUMBER OF RECORDS TO BE ON EACH FILE'
      SET &CNT TO &PTEREPLY
      GOTO PARM000
   ELSE
      COMPUTE &REC-CNT = &CNT
   ENDIF
PARM010 IF &GENS EQ ''
   THEN
      WTOR 'ENTER NUMBER OF GENERATIONS TO BE BUILT'
      SET &GENS TO &PTEREPLY
      GOTO PARM010
   ENDIF
PARM020 IF &GDGBASE EQ ''
   THEN
      WTOR 'ENTER BASE NAME OF GDG TO BE BUILT'
      SET &GDGBASE TO &PTEREPLY
   ENDIF
   SET &ENDFILE-MSG TO 'ENTER "END" OR "REUSE" FOR ACTION TO '
                       'BE TAKEN AT ENDFILE ON SYSUT1 FILE'
PARM030 IF &ENDFILE EQ 'REUSE'
   THEN
      NEXT
   ELSE
      IF &ENDFILE EQ 'END' OR &ENDFILE EQ ''
   THEN
      SET &ENDFILE TO 'END'
   ELSE
      WTOR &ENDFILE-MSG
      SET &ENDFILE TO &PTEREPLY
      GOTO PARM030
   ENDIF
      ENDIF
PARM040 IF &ORDER EQ 'RAN'
   THEN
      NEXT
   ELSE
      IF &ORDER EQ 'SEQ' OR &ORDER EQ ''
   THEN
      SET &ORDER TO 'SEQ'
   ELSE
      WTOR 'ENTER "SEQ" OR "RAN" FOR RECORD ORDER'
```

```
      SET &ORDER TO &PTEREPLY
      GOTO PARM040
   ENDIF
      ENDIF
PARM050 IF &FREQ EQ ''
   THEN
      SET &FREQ TO 1
   ELSE
      IF &FREQ LT 1
   THEN
      WTOR 'ENTER "FREQ" INTEGER VALUE GREATER THAN 0'
      SET &FREQ TO &PTEREPLY
      GOTO PARM050
   ENDIF
      ENDIF
   OPEN INPUT SYSUT1 /* INSURE FILE HAS RECORDS */
   READ SYSUT1 INTO &INPUT ERROR *
   IF &PTERESP NE '0000'
   THEN
      LOG 'ERROR ATTEMPTING TO READ FIRST RECORD ON SYSUT1'
      ABEND 4095
   ENDIF
   CLOSE SYSUT1
   SET &GDG TO 0
   DO WHILE &GDG LT &GENS
      SET &OUTPUT-DSN(&GDG) TO &GDGBASE '(-' &GDG ')'
      ALLOCATE &DD(&GDG) DATASET &OUTPUT-DSN(&GDG) SHR ERROR *
      IF &PTERESP NE '0000'
         THEN
            LOG 'ALLOCATE ERROR-' &OUTPUT-DSN(&GDG) ' RC=' &PTERESP
            ABEND
         ELSE
            OPEN OUTPUT &DD(&GDG)
      ENDIF
      COMPUTE &GDG = &GDG + 1
   ENDDO
   PERFORM SYSUT1-OPEN
   IF &ORDER EQ 'SEQ'
      THEN
         SET &GDG TO 0
         DO WHILE &GDG LT &GENS
            SET &REC-CNT TO 0
            DO WHILE &REC-CNT LT &CNT
               PERFORM COMMON-ROUTINE
               SET &REC-CNT UP 1
      ENDDO
      SET &GDG UP 1
   ENDDO
   ELSE
      SET &REC-CNT TO 0
```

```
      DO WHILE &REC-CNT LT &CNT
         SET &GDG TO 0
            DO WHILE &GDG LT &GENS
            PERFORM COMMON-ROUTINE
         SET &GDG UP 1
         ENDDO
      SET &REC-CNT UP 1
      ENDDO
   ENDIF
EOJ IF &SYSUT1-IND EQ 'OPEN'
   THEN
      CLOSE SYSUT1
ENDIF
SET &GDG TO 0
DO WHILE &GDG LT &GENS
CLOSE &DD(&GDG)
FREE &DD(&GDG)
SET &GDG UP 1
ENDDO
LOG &RECORDS-SELECTED ' RECORDS SELECTED FROM INPUT FILE'
LOG &OUTPUT-REC-CNT ' RECORDS WRITTEN TO ' &GENS ' FILE(S)'
TIME
LOG 'SCRIPT ENDED AT ' &PTEHR ':' &PTEMIN ':' &PTESEC
PAUSE
END
COMMON-ROUTINE
PERFORM SYSUT1-READ
LINK &FORMAT /* LINK TO USER SCRIPT */
IF &OUTPUT-RECORD EQ '' /* IF USER SCRIPT DIDN'T FORMAT */
THEN /* RECORD USE INPUT RECORD */
SET &OUTPUT-RECORD TO &INPUT-RECORD
ENDIF
WRITE &DD(&GDG) FROM &OUTPUT-RECORD
SET &OUTPUT-REC-CNT UP 1
DROP &OUTPUT-RECORD
EXIT
SYSUT1-READ
READ SYSUT1 INTO &INPUT-RECORD ENDFILE SYSUT1-EOF
SET &SEL UP 1
IF &SEL LT &FREQ /* ONLY SELECT EVERY N'TH RECORD */
THEN
GOTO SYSUT1-READ
ENDIF
SET &SEL TO 0
SET &RECORDS-SELECTED UP 1
EXIT
SYSUT1-EOF
CLOSE SYSUT1
SET &SYSUT1-IND TO 'CLOSED'
IF &ENDFILE EQ 'REUSE'
```

```
THEN
PERFORM SYSUT1-OPEN
GOTO SYSUT1-READ
ELSE
GOTO EOJ
ENDIF
SYSUT1-OPEN
OPEN INPUT SYSUT1
SET &SYSUT1-IND TO 'OPEN'
EXIT
```

The CA Enterprise Tester Session Analyzer

A session analyzer is a CA Enterprise Tester script that reads screens from a previously created CA Enterprise Tester trace data sets, re-keys input on a new live session, and compares live online system responses to traced online system responses to detect differences. Session analyzers may be executed using the Interactive System Analyze or Play panels or may be executed using a batch job. The distributed session analyzers, PTESA, is provided in the distributed CA Enterprise Tester script library, and is documented in The CA Enterprise Tester Session Analyzer. This section summarizes the logic of the distributed session analyzer.

While the distributed analyzer is a general purpose solution to that may be used to analyze sessions with any online system, customized session analyzers that take advantage of knowledge of applications, may be used to solve special problems and provide additional benefits. For example:

- If modifications made to an online system intentionally modify the response to certain requests, then a session analyzer that compares full screens would produce predictable mismatches. Using all of the logic and communication capabilities provided by the CA Enterprise Tester Script Language, a customized session analyzer could examine responses and automatically use special logic to compare selected screens, to enter extra transactions needed with upgraded online software, or to discard screens no longer supported by upgraded software.
- If online application input fields are moved to different screen locations, then not only would the distributed session analyzer produce predictable mismatches, but also would not know to re-key traced input at a new screen location. A customized session analyzer could examine traced and live session screens and re-key traced input at any desired screen location automatically.

PTESA exploits PSL statements that were designed to support applications like session analyzers. The TRACK statement allows CA Enterprise Tester scripts to read an entire screen image and related information from a trace data set using a single PSL statement. When TRACKED data is keyed into a virtual terminal, built in TYPE statement logic knows to key only modified data. Session analyzer masking capabilities are based on the MASK statement and supporting logic that is built into the PSL IF statement. Using the built in language features designed to support the manipulation and processing of traced and virtual terminal sessions, you may create customized session analyzers that automatically deal with situations unique to your environment or applications.

The following summary of the logic of PTESA, and the PTESA listing that follows, illustrates how PSL statements may be used to develop high-level session analyzers.

- PTESA supports many options that may be specified as JCL or Play Panel parameters, as PSL variables that are set by scripts that LINK to PTESA, or as control statements of any desired format that are stored in a dataset. The PTESA LOAD routine processes analyzer options from any source using PSL statements to parse and save options and data sets to be processed, including analyzer options that may be supplied in 'trace list'format defined by PTESA.
- PTESA uses PSL DO loops, PERFORM and LINK statements to structure analyzer logic in a easy-to-understand and maintainable format. The mainline loop performs a routine named STEP to get a session step to be analyzed then performs one of two routines, COMPARE and REKEY, to process each screen image returned by STEP, depending on whether the screen image is an input screen to be repeated or a response screen to be compared to live session response to a repeated transaction. PSL LINK statements are also used to invoke other scripts that may be specified by PTESA users to customize analyzer logic.
- The PTESA STEP routine uses PSL variables returned by the TRACK statement to analyze data returned by the TRACK routine, and uses the data to start new virtual terminal sessions using PSL LOGON, LINK or EXECUTE statements, and uses PSL INVITE and DELAY statements to synchronize the session with traced session data. The STEP routine returns either an input screen to be rekeyed on the live session or returns a response screen to be compared to the live virtual terminal session display.

- The PTESA REKEY routine uses special PSL TYPE statement features to rekey input data from an inbound TRACKED screen image into the live virtual terminal display, and uses other PSL statements to analyze and report any problems that may occur if online transactions responses are not similar to the previously traced responses. :li, The PTESA COMPARE routine uses to the PSL LINK statement to invoke user supplied scripts that use the PSL MASK statement to identify predictable response differences like date or time, and relies on related logic built into the PSL IF statement to automatically suppress detection of the identified differences when TRACKED data is compared to the live session data. When unmasked differences are found, the compare routine uses PSL SET IMAGE, ATTRIBUTES and other PSL capabilities to determine and report precisely what differences were detected.
- The PTESA OPEN routine is invoked to process each trace data set to be processed and to set up analyzer options to be used to process each trace data set. OPEN uses special PSL variable, PTEPGM, to tailor processing for batch or interactive use, and uses PSL TRACE statements to automatically trace live session results that may be used to upgrade baseline trace dataset members.
- The PTESA TRACK routine uses the PSL TRACK statement to read entire screen images from trace datasets or to read entire PSL statements that were captured when a trace data set was created, and uses other general purpose PSL statements to CLOSE, FREE and ENDTRACE statements to manage data sets.

The following listing is the distributed session analyzer, PTESA:

```
*-------------------------------------------------------------------*
* PTESA is the distributed ProTerm Session Analyzer. *
*-------------------------------------------------------------------*
* PTESA repeats traced sessions using a new ProTerm virtual terminal *
* session, compares old responses to new responses and reports any *
* differences found. *
* *
* PTESA may be started using the Analyze or Play panels, may be *
* linked to by a script that sets analyzer options, or may be played *
* in batch. *
* *
* PTESA processes either a Format trace data set or a trace list that *
* contains ANALYZE statements that specify trace data sets to be *
* processed, and may also specify analyzer options, using the *
* analyzer option variable name as a trace list keyword. *
* *
* The data set to be processed is specified in TRACE or TRACE and *
* MEMBER option variables that may be assigned values using script *
* parameters or using SET statements in a script that links to PTESA. *
* A TRACE DD may also be used to supply the input data set when no *
* value is supplied in the TRACE option variable. *
* *
* The following trace list data set may be specified via the Analyze *
```

```
* Panel to analyze the specified trace data sets. *
* *
* MASK=mymask1 *
* ANALYZE=my.trace.library(add) *
* ANALYZE=my.trace.library(update) *
* MASK=mymask2 *
* ANALYZE=my.trace.library(delete) *
* *
* The following script may be specified at the Analyze Panel as an *
* Analyzer Script to process a specified trace or trace list dataset. *
* *
* set &delays to 'time' *
* set &time to '0500' *
* set &logon to 'mylogon' *
* set &logoff to 'mylogoff' *
* set &tracepds to 'my.upgrade.trace' *
* link 'ptesa' *
* *
* Sample PTESA JCL: *
* *
* //PTESA EXEC PGM=PTEBS,PARM='TRACE=UID.TRACE(MEM1),MASK=MASK1' *
* //PTEIN DD DISP=SHR,DSN=SYS.PROTERM.SCRIPT(PTESAFE) *
* //PTELIB DD DISP=SHR,DSN=UID.PROTERM.SCRIPT *
* //PTELOG DD DISP=SHR,DSN=UID.PROTERM.LOG(MEM1) *
* //PTETRACE DD DISP=SHR,DSN=UID.PROTERM.TRACE(MEM1) *
* *
* PTESA Options: *
* *
* Variable Value *
* -------- ------------------------------------------------------------ *
* *
* DELAYS 'Synctime' instructs PTESA to use the PSL INVITE statement *
* and the time specified using the SYNCTIME analyzer option *
* only to receive anticipated consecutive outbound messages *
* implied by consecutive outbound messages found in trace *
* data. The option value is appropriate only for sessions *
* that send a predictable number of responses to each *
* transaction entered. *
* *
* 'Time' instructs PTESA to use the PSL DELAY statement and *
* a time value specified using the TIME analyzer option to *
* receive possible delayed responses that may follow an *
* outbound message that leaves the analyzer session in the *
* CONTENTION state. Either DELAYS 'Time' or the 'Bracket' *
* option should be used to synchronize the analyzer session *
* with the traced session when testing systems that may *
* respond unpredictably. *
* *
* 'Bracket' instructs PTESA to execute PSL DELAY statements *
* found in traced session data to receive possible delayed *
```

```
* responses that may follow an outbound message that leaves *
* the analyzer session in the CONTENTION state. If DELAY *
* statements are not found in traced session data, a DELAY *
* is issued that uses the time specified by the TIME *
* analyzer option. *
* *
* 'Input' instructs PTESA to execute all PSL DELAY *
* statements found in trace data even if there is no *
* possibility that the online system may send a message or *
* to issue a DELAY statement using the time specified by the *
* TIME analyzer option if DELAY statements are not found in *
* traced session data. *
* *
* SYNCTIME Is a time in the form SSTH, that is to be used as an *
* INVITE time when the DELAYS 'Synctime' option is active *
* and consecutive outbound messages are expected on a live *
* session that correspond to consecutive outbound messages *
* found in a trace data set being analyzed. The default *
* value is 1000 (10 seconds). *
* *
* TIME Is a time in the form SSTH, to be used as a PSL DELAY *
* statement time when the DELAYS 'Time' option is active, or *
* when the DELAYS Bracket or 'Input' option is active and a *
* PSL DELAY statement is not found in a trace data set, to *
* receive possible delayed responses that may be received *
* following a response that leaves the analyzer session in *
* the CONTENTION state. *
* *
* DELAYPCT Is a percent from 0 to 999 that is a percent of the time *
* specified by PSL DELAY statement found in trace data sets, *
* when the DELAY statements are executed by PTESA while *
* either the DELAYS 'Bracket' or 'Input' option is active. *
* *
* IMAGE Normally, PTESA compares all corresponding data contained *
* in trace and live session screens, including data *
* contained in non-display screen fields, as binary values. *
* IMAGE may be set to 'Yes' to request that the image of *
* screen data be compared. The image of screen field data *
* is what a person sees on a screen. When IMAGE is set to *
* 'Yes', data in non-display screen fields and binary zeros *
* are considered to be blanks when screen data is compared. *
* *
* CASE By default, PTESA is sensitive to differences in the case *
* of alphabetic characters that are compared. CASE may be *
* set to 'No' to specify that corresponding upper and lower *
* case alphabetic character values are to be considered *
* equal when they are compared. *
* *
* ATTR By default, PTESA detects and documents any differences in *
* corresponding screen field attributes. ATTR may be set to *
```

```
* 'No' to specify that field attributes at corresponding *
* trace and live session screen locations are to be *
* considered equal. *
* *
* LOGON Is the name of a script that is to be linked to before *
* analyzing traced screens. A LOGON script should start a *
* session to be used to analyze traced screens and should *
* enter sign on or other transactions needed to position the *
* session at the screen that matches the first traced screen *
* to be analyzed. *
* *
* MASK Is the name of a script that is to be linked to before *
* comparing each pair of subsequently analyzed trace and *
* live session screens. A MASK script may examine the *
* screens and execute PSL MASK statements to suppress the *
* comparison of insignificant data like date and time *
* displays. *
* *
* TYPE Is the name of a script that is to be linked to after *
* TRACKED data has been rekeyed but prior to the AID key *
* being typed. A TYPE script may modify or supplement data *
* rekeyed from an old traced screen image when application *
* input fields are moved or added. Upon return from the *
* script, the AID key from the traced session is typed. *
* *
* EDIT Is the the name of a script that is to be linked to by *
* PTESA prior to comparing a TRACKED screen image to the *
* current virtual terminal display. An EDIT script may *
* manipulate the trace dataset and/or the live session if *
* the script determines that a correctable out-of-sync *
* condition exists. *
* *
* FILTER Is the name of a script that is to be linked to after *
* starting a session and when the terminal keyboard is freed *
* following each re-keyed terminal input. The script may *
* examine the traced input screen, the live session screen, *
* and may issue INVITE or TYPE statements to manipulate the *
* live session in any desired way. *
* *
* LINK Is the name of a script that is to be linked to prior to *
* the processing the trace data set named by the subsequent *
* ANALYZE statement. Multiple LINK statements preceeding *
* an ANALYZE statement are linked to in the order they are *
* listed. A LINK statement is processed before processing *
* the subsequent ANALYZE statement only and is not processed *
* before processing any other ANALYZE statement. *
* *
* SYNC Is the name of a script that is to be linked to after *
* processing a trace data sets named by a trace list ANALYZE *
* statement. After a trace data set is processed, a *
```

```
* previously specified SYNC script may verify that a session *
* has returned to a "primary menu", that is the assumed *
* starting point of session activity contained in *
* subsequently processed trace data sets. If an intended *
* menu is not displayed, a SYNC script may enter *
* transactions needed to "return" to the menu, allowing *
* subsequent trace data sets to process correctly. *
* *
* LOGOFF Is the name of a script that is to be linked to after *
* processing all trace data being analyzed. A LOGOFF script *
* may execute any transactions needed to end a session in an *
* orderly manner. *
* *
* TRACEPDS Is the name of a trace library to be used to capture new *
* sessions created by repeating transactions found in the *
* old trace data being analyzed. When a TRACEPDS is *
* specified, a new member is created in the specified *
* TRACEPDS that is named the same as each old trace member *
* analyzed. Each new TRACEPDS member is opened and closed *
* at the same time as the input trace member to create *
* upgraded trace members that contain session events *
* equivalent to the events in the old trace members. *
* *
* SCREENS Specifies the outbound screens to capture in a specified *
* TRACEPDS. 'Freekb' specifies that outbound screen images *
* are to be traced only when messages are received that free *
* the terminal keyboard. 'Chains' specifies that outbound *
* screen images are to traced when any complete outbound *
* message is received. *
* *
* MISMATCH Is the action taken when response mismatches are detected *
* by PTESA. The default MISMATCH action is 'Pause'. *
* *
* 'Continue' instructs PTESA to continue processing without *
* interruption after documenting each response mismatch. *
* *
* 'Pause' instructs PTESA to stop and wait for terminal user *
* action after documenting each response mismatch. Because *
* PTESA can not communicate with a terminal user when it is *
* running in batch, 'Pause' is changed to 'Continue' when *
* PTESA is used to perform regression testing in batch. *
* *
* 'Stop' instructs PTESA to stop processing after *
* documenting a response mismatch. When PTESA is used to *
* perform regression testing interactively, 'Stop' is *
* changed to 'Pause' allowing the terminal user to decide *
* whether to continue or end processing. *
* *
* KEYERROR Is action to be taken when the analyzer can not rekey *
* trace input data into the live session screen. The *
```

```
* default KEYERROR action is 'Stop'. *
* *
* 'Continue' instructs PTESA to document the problem but to *
* continue processing by entering the transaction even *
* though all input data was not rekeyed. *
* *
* 'Pause' instructs PTESA to stop and wait for terminal user *
* action after documenting the problem. Because PTESA can *
* not communicate with a terminal user when it is running in *
* batch, 'Pause' is changed to 'Continue' when PTESA is used *
* to perform regression testing in batch. When PTESA us *
* used interactively and pauses because it is unable to *
* rekey traced input data, the transaction is not entered. *
* The terminal user may select the live session, and may key *
* in any data and enter the transaction manually. After *
* completing a transaction manually, PTESA may be resumed to *
* repeat the next traced transaction. *
* *
* 'Stop' instructs PTESA to stop processing after *
* documenting the problem. When PTESA is used to perform *
* regression testing interactively, 'Stop' is changed to *
* 'Pause' allowing the terminal user to complete the *
* transaction manually and to decide whether to continue or *
* end processing. *
* *
*------------------------------------------------------------------------*
* Copyright (C) 2008, by Reed Software, Inc. *
*------------------------------------------------------------------------*
perform load
do forever
perform step
if &eof = 'yes'
then
leave
endif
if &ptetdir = 'OUT'
then
if &edit > ''
then
link &edit
endif
if &ptestate \= 'OFFLINE'
then
perform compare
endif
else
perform rekey
endif
enddo
go to return
```

```
*--------------------------------------------------------------------*
* STEP selects next session step and synchronizes trace and session. *
*--------------------------------------------------------------------*
step
do
perform track
if &eof = 'yes'
then
exit
endif
if &pteresp = 'LINE'
and &ptetline(,1,5) contains 'logon'
then
set &auto-logon to 'no'
set &t-responses to 0
set &s-responses to 0
log 'Executing: ' &ptetline
execute &ptetline
if &ptestate = 'OFFLINE'
then
set &mess to 'Executed LOGON failed. Processing ended.'
go to close
endif
set &sessions up 1
iterate
endif
if &ptestate = 'OFFLINE'
then
if &auto-logon = 'ok'
then
set &auto-logon to 'no'
set &t-responses to 0
set &s-responses to 0
if &ptetplu = ''
then
set &mess to
'Unable to start a session; No system name.'
go to close
endif
log 'Trace title logon: LOGON ' &ptetplu
' MODEL ' &ptetmodl ' ' &ptetluds
' ' &ptetdsiz ' ' &ptetasiz
logon &ptetplu model &ptetmodl
&ptetluds &ptetdsiz &ptetasiz
if &ptestate = 'OFFLINE'
then
set &mess to 'Logon failed. Processing ended.'
go to close
endif
set &sessions up 1
```

```
else
if &pteresp = 'LINE' or &ptetdir \= 'OUT'
then
set &mess to 'Session Lost. Processing ended.'
go to close
endif
iterate
endif
endif
if &pteresp = 'LINE'
then
if &ptetline(,1,5) contains 'delay'
then
if &ptetline(,7) is decimal
then
trace delays
endif
else
log 'Executing: ' &ptetline
execute &ptetline
endif
iterate
endif
if &ptetseq = '#00000'
then
set &trace-seq to &ptettod
else
set &trace-seq to &ptetseq
endif
if &ptetdir = 'OUT'
then
if &delays = 'Synctime'
then
set &t-responses up 1
do
if &t-responses = 1
then
set &trace-seq1 to &trace-seq
if &ptestate \= 'ONLINE'
then
set &s-responses to 1
leave
endif
else
if &ptestate \= 'CONTENTION'
then
leave
endif
endif
invite &synctime
```

```
if &ptestate = 'OFFLINE'
then
iterate
endif
if &ptecc > 0
then
set &s-responses up 1
endif
enddo
if &ptetnext = 'OUT'
then
iterate
endif
else
do
if &delays \= 'Time'
and &ptetnext(,1,5) contains 'delay'
then
perform track
if &eof = 'yes'
then
exit
endif
if &ptetline(,7) is decimal
then
trace delays
execute &ptetline
leave
endif
endif
delay &time
enddo
if &filter \= ''
and (&ptestate = 'CONTENTION'
or &ptestate = 'ONLINE')
then
link &filter
endif
if &ptestate = 'OFFLINE'
then
iterate
endif
endif
else
set &s-responses to 0
set &t-responses to 0
if &caption ne &ptetcapt
then
set &caption to &ptetcapt
record ' TRACE "' &caption '"'
```

```
endif
if &ptestate = 'ONLINE'
then
if &delays = 'Synctime'
then
invite &synctime
else
delay &time
if &ptestate = 'CONTENTION'
or &ptestate = 'ONLINE'
then
link &filter
endif
endif
if &ptestate = 'OFFLINE'
then
set &mess to 'Session Lost. Processing ended.'
go to close
endif
endif
endif
enddo
exit
*-------------------------------------------------------------------------*
* REKEY rekeys data from an inbound screen image returned by STEP. *
*-------------------------------------------------------------------------*
* Rekey Messages *
* -------------- *
* *
* Trace data at <r c> could Not be rekeyed in &attr field.
* Row 3 input +----+----1----+----2----+- More: Scroll "Right 21" -->
* Not rekeyed | NN N
* Source row | TSS7000I RS6 Last-Used 25 Sep 96 09:25 System=FR
* Target row | TSS7000I RS6 Last-Used 25 Sep 96 14:22 System=FR
*
* Row 3 input Not rekeyed, source row, and target row:
* +----+----1----+----2----+----3----+----4- More: Scroll "Right 58" -->
* | NN N
* | TSS7000I RS6 Last-Used 25 Sep 96 09:25 System=FR
* | TSS7000I RS6 Last-Used 25 Sep 96 14:22 System=FR
*
*-------------------------------------------------------------------------*
rekey
set &rekeys up 1
set &messages to 0
set &action to &keyerror
if &ptetsize \= &ptesize
then
set &mess(1) to 'Trace data from screen ' &trace-seq
' in ' &dataset
```

```
set &mess(2) to 'was not rekeyed as input to transaction '
&rekeys ', because the sizes of the'
set &mess(3) to 'trace and session screens, '
&ptetrows 'X' &ptetcols ' and ' &pterows 'X' &ptecols
', do not match.'
set &messages to 3
else
set &r to 0
do while &r < &pterows
set &r up 1
set &flag-row(,1,132) to all ' '
set &c to 1
do while &c le &ptetcols
compute &kel = &ptetcols - &c + 1
type <&r &c> tracked <&r &c &kel>
if &pteke eq 0
then
leave
endif
compute &c = &c + &pteke
if &flag-row = ' '
then
perform page
endif
compute &flag-col = &c - 1
set &flag-row(,&flag-col,1) to 'X'
set &tattr to attributes <&r &flag-col 1>
do while &c le &ptetcols
set &sattr to attributes <&r &c 1>
if &sattr = &tattr
then
set &flag-row(,&c,1) to 'X'
set &rightmost-col to &c
set &c up 1
else
leave
endif
enddo
set &messages up 1
set &mess(&messages) to
'Trace data at <' &r ' ' &flag-col
'> could not be rekeyed in ' &tattr ' field'
enddo
if &flag-row \= ' '
then
perform page
if &ptecols > 80
then
set &tdata to image tracked <&r 1 132>
set &sdata to image <&r 1 132>
```

```
set &messages up 1
set &mess(&messages) to 'Row ' &r
' input not rekeyed, source row, and target row:'
set &messages up 1
set &mess(&messages) to '+' &hscale(,1,132) '+'
if &keyerror = 'Pause' and &rightmost-col > 75
then
set &mess(&messages,49,29) to
' More: Scroll "Right 58" --> '
endif
set &messages up 1
set &mess(&messages) to '|' &flag-row(,1,132) '|'
set &messages up 1
set &mess(&messages) to '|' &tdata '|'
set &messages up 1
set &mess(&messages) to '|' &sdata '|'
else
set &tdata to image tracked <&r 1 80>
set &sdata to image <&r 1 80>
set &messages up 1
set &row-label(,1,15) to 'Row ' &r ' input'
set &mess(&messages) to
&row-label '+' &hscale(,1,80) '+'
if &keyerror = 'Pause' and &rightmost-col > 60
then
set &mess(&messages,49,29) to
' More: Scroll "Right 21" --> '
endif
set &messages up 1
set &mess(&messages) to
' Not rekeyed |' &flag-row(,1,80) '|'
set &messages up 1
set &mess(&messages) to
' Source row |' &tdata '|'
set &messages up 1
set &mess(&messages) to
' Target row |' &sdata '|'
endif
endif
enddo
endif
link &type /* allow user script to do typing */
type <&ptetrow &ptetcol>
set &inbuff to image <1 1 &ptesize>
set &inrows to &pterows
set &incols to &ptecols
set &inkey to &ptetaid
set &inseq to &trace-seq
if &messages > 0
then
```

```
set &keyerrors up 1
log &dashes
log 'Transaction ' &rekeys
' input could not be rekeyed.'
log ' '
log ' Transaction ' &rekeys ' input was screen '
&trace-seq ' in ' &dataset ':'
log '+' &hscale(,1,&ptetcols) '+'
set &display to image tracked <1 1 &ptetsize>
set &p to 1
set &r to 0
do while &r < &ptetrows
set &r up 1
log &vscale(,&r,1) &display(,&p,&ptetcols) &vscale(,&r,1)
set &p up &ptetcols
enddo
log '+' &hscale(,1,&ptetcols) '+'
log ' '
log ' The transaction ' &rekeys
' session display after rekeying all possible input:'
log '+' &hscale(,1,&ptecols) '+'
set &display to image <1 1 &ptesize>
set &p to 1
set &r to 0
do while &r < &pterows
set &r up 1
log &vscale(,&r,1) &display(,&p,&ptecols) &vscale(,&r,1)
set &p up &ptecols
enddo
log '+' &hscale(,1,&ptecols) '+'
log ' '
set &m to 0
do while &m < &messages
set &m up 1
log &mess(&m)
enddo
if &keyerror = 'Stop'
then
log ' '
set &mess to 'Analyzer ended on Keyerror=Stop'
go to close
endif
if &keyerror = 'Pause'
then
compute &log-lines = &messages / remainder 8
if &log-lines > 0
then
do while &log-lines < 5
set &log-lines up 1
log ' '
```

```
enddo
endif
log 'REVIEW, then ROLL to visually compare full screen'
' displays.'
log 'Select session, complete transaction manually,'
' then STEP or VIEW to repeat next transaction.'
pause
log ' '
do until &ptetnext = 'IN'
or &ptetnext = 'EOF'
or &ptetnext = ''
perform track
enddo
exit
else
log ' '
log 'Transaction ' &rekeys ' processing continued.'
endif
log ' '
endif
type aid &ptetaid
exit
*-------------------------------------------------------------------------*
* COMPARE compares a tracked screen image to the live session display *
*-------------------------------------------------------------------------*
* Mismatch messages *
* ----------------- *
*
* Trace <r 1> Attributes: FIELD PROTECTED
* Session <r 1> Attributes: CHARACTER MODIFIABLE
*
* Row 3 Non-display data differences:
* Old hex zone | 0 0 00000
* numeric | 0 0 00000
* +----+----1----+----2----+- More: Scroll "Right 21" -->
* New hex zone | 4 4 44444
* numeric | 0 0 00000
*
* Row 3 Display +----+----1----+----2----+- More: Scroll "Right 21" -->
* differences |A N N DD D NNNNN
* old display | TSS7000I RS6 Last-Used 25 Sep 96 09:25 System=FR
* new display | TSS7000I RS6 Last-Used 25 Sep 96 14:22 System=FR
*
*
* Row 3 Non-display data differences; old and new hexadecimal values:
* | 0 0 00000000000000000000
* | 0 0 00000000000000000000
* +----+----1----+----2----+----3----+----4- More: Scroll "Right 58" -->
* | 4 4 44444444444444444444
* | 0 0 00000000000000000000
```

```
*
* Row 3 display differences, old display, and new display:
* +----+----1----+----2----+----3----+----4- More: Scroll "Right 58" -->
* |A N N DD D NNNNNNNNNNNNNNNNNNN
* | TSS7000I RS6 Last-Used 25 Sep 96 09:25 System=FR
* | TSS7000I RS6 Last-Used 25 Sep 96 14:22 System=FR
*
*-------------------------------------------------------------------------*
compare
set &compares up 1
set &messages to 0
set &action to &mismatch
if &t-responses \= &s-responses
then
set &mess(1) to &t-responses
' responses, that correspond to screens ' &trace-seq1
' through ' &trace-seq ','
set &mess(2) to 'in ' &dataset ', were expected,'
set &mess(3) to 'but only ' &s-responses ' were received. '
' Increase SYNCTIME=' &synctime ', or use'
set &mess(4) to 'DELAYS=TIME or DELAYS=BRACKET'
' to test systems that unpredictably queue'
set &mess(5) to 'requests, send delayed responses,'
' or send a variable number of responses.'
set &messages to 5
endif
do
if &ptetsize \= &ptesize
then
perform page
set &messages up 1
set &mess(&messages) to 'Trace response screen ' &trace-seq
' in ' &dataset
set &messages up 1
set &mess(&messages) to
'was not compared to the transaction ' &rekeys
' response, because the sizes of'
set &messages up 1
set &mess(&messages) to 'the trace and session screens, '
&ptetrows 'X' &ptetcols ' and ' &pterows 'X' &ptecols
', do not match.'
leave
endif
if &pterow ne &ptetrow or &ptecol ne &ptetcol
then
perform page
set &messages up 1
set &mess(&messages) to
'The trace and session cursor locations, <'
&ptetrow ' ' &ptetcol '> and <' &pterow ' ' &ptecol
```

```
'>, do not match.'
endif
link &mask
if <1 1 &ptesize> eq tracked <1 1 &ptetsize>
then
leave
endif
set &r to 0
do while &r < &pterows
set &r up 1
if <&r 1 &ptecols> eq tracked <&r 1 &ptecols>
then
iterate
endif
set &flag-row(,1,132) to all ' '
set &c to 1
do while &c le &ptecols
compute &nel = &ptecols + 1 - &c
if <&r &c &nel> = tracked <&r &c &nel>
then
leave
endif
compute &c = &ptecols + 1 - &ptenel
do
if &attr = 'No'
then
set &tattr to attributes tracked <&r &c>
set &sattr to attributes <&r &c>
if &sattr contains 'field' and
&tattr contains 'field'
then
leave
endif
endif
if &image = 'Yes'
then
set &simag to image <&r &c 1>
set &timag to image tracked <&r &c 1>
if &simag = &timag
then
leave
endif
endif
if &case = 'No'
then
set &suppr to capital <&r &c 1>
set &tuppr to capital tracked <&r &c 1>
if &suppr = &tuppr
then
leave
```

```
endif
endif
if &flag-row = ' '
then
set &flag-tz to &flag-row
set &flag-tn to &flag-row
set &flag-sz to &flag-row
set &flag-sn to &flag-row
perform page
endif
set &flag-row(,&c,1) to 'D'
set &rightmost-col to &c
set &tattr to attributes tracked <&r &c 1>
set &sattr to attributes <&r &c 1>
if &tattr \= &sattr
then
set &flag-row(,&c,1) to 'A'
set &messages up 1
set &mess(&messages) to 'Old row ' &r ' col ' &c
' Attributes: ' &tattr
set &messages up 1
set &mess(&messages) to 'New row ' &r ' col ' &c
' Attributes: ' &sattr
else
if &tattr contains 'character'
then
set &timag to image tracked <&r &c 1>
set &simag to image <&r &c 1>
if &timag = &simag
then
set &flag-row(,&c,1) to 'N'
set &rightmost-hex to &c
set &hex from hex tracked <&r &c 1>
set &flag-tz(,&c,1) to &hex(,1,1)
set &flag-tn(,&c,1) to &hex(,2,1)
set &hex from hex <&r &c 1>
set &flag-sz(,&c,1) to &hex(,1,1)
set &flag-sn(,&c,1) to &hex(,2,1)
endif
endif
endif
enddo
set &c up 1
enddo
if &flag-row \= ' '
then
perform page
if &ptecols > 80
then
if &flag-tz \= ' '
```

```
then
set &messages up 1
set &mess(&messages) to
'Row ' &r 'Non-display data differences;'
' old and new hexadecimal values:'
set &messages up 1
set &mess(&messages) to '|' &flag-tz(,1,132) '|'
set &messages up 1
set &mess(&messages) to '|' &flag-tn(,1,132) '|'
set &messages up 1
set &mess(&messages) to '+' &hscale(,1,132) '+'
if &mismatch = 'Pause' and &rightmost-hex > 75
then
set &mess(&messages,49,29) to
' More: Scroll "Right 58" --> '
endif
set &messages up 1
set &mess(&messages) to '|' &flag-sz(,1,132) '|'
set &messages up 1
set &mess(&messages) to '|' &flag-sn(,1,132) '|'
perform page
endif
set &tdata to image tracked <&r 1 132>
set &sdata to image <&r 1 132>
set &messages up 1
set &mess(&messages) to 'Row ' &r ' Display'
' differences, old display, and new display:'
set &messages up 1
set &mess(&messages) to '+' &hscale(,1,132) '+'
if &mismatch = 'Pause' and &rightmost-col > 75
then
set &mess(&messages,49,29) to
' More: Scroll "Right 58" --> '
endif
set &messages up 1
set &mess(&messages) to '|' &flag-row(,1,132) '|'
set &messages up 1
set &mess(&messages) to '|' &tdata '|'
set &messages up 1
set &mess(&messages) to '|' &sdata '|'
else
if &flag-tz \= ' '
then
set &messages up 1
set &mess(&messages) to 'Row ' &r
' Non-display data differences:'
set &messages up 1
set &mess(&messages) to
'old hex zone |' &flag-tz(,1,80) '|'
set &messages up 1
```

```
set &mess(&messages) to
' numeric |' &flag-tn(,1,80) '|'
set &messages up 1
set &mess(&messages) to
' +' &hscale(,1,80) '+'
if &mismatch = 'Pause' and &rightmost-hex > 60
then
set &mess(&messages,49,29) to
' More: Scroll "Right 21" --> '
endif
set &messages up 1
set &mess(&messages) to
'new hex zone |' &flag-sz(,1,80) '|'
set &messages up 1
set &mess(&messages) to
' numeric |' &flag-sn(,1,80) '|'
perform page
endif
set &tdata to image tracked <&r 1 80>
set &sdata to image <&r 1 80>
set &messages up 1
set &row-label(,1,15) to 'Row ' &r ' Display'
set &mess(&messages) to
&row-label '+' &hscale(,1,80) '+'
if &mismatch = 'Pause' and &rightmost-col > 60
then
set &mess(&messages,49,29) to
' More: Scroll "Right 21" --> '
endif
set &messages up 1
set &mess(&messages) to
' differences |' &flag-row(,1,80) '|'
set &messages up 1
set &mess(&messages) to
' old display |' &tdata '|'
set &messages up 1
set &mess(&messages) to
' new display |' &sdata '|'
endif
endif
enddo
enddo
mask reset
if &messages > 0
then
set &mismatches up 1
log &dashes
log 'Transaction ' &rekeys
' response did not match previously traced response.'
log ' '
```

```
if &rekeys > 0
then
log ' Transaction ' &rekeys
' input rekeyed from screen ' &inseq
' in ' &dataset ':'
log '+' &hscale(,1,&incols) '+'
set &p to 1
set &r to 0
do while &r < &inrows
set &r up 1
log &vscale(,&r,1) &inbuff(,&p,&incols) &vscale(,&r,1)
set &p up &incols
enddo
log '+' &hscale(,1,&incols) '+'
log ' '
log ' New response received when ' &inkey
' was pressed to send rekeyed input:'
else
log ' New response received when session was started:'
endif
log '+' &hscale(,1,&ptecols) '+'
set &display to image <1 1 &ptesize>
set &p to 1
set &r to 0
do while &r < &pterows
set &r up 1
log &vscale(,&r,1) &display(,&p,&ptecols) &vscale(,&r,1)
set &p up &ptecols
enddo
log '+' &hscale(,1,&ptecols) '+'
log ' '
log ' Old response received when screen ' &trace-seq
' in ' &dataset ' was traced:'
log '+' &hscale(,1,&ptetcols) '+'
set &display to image tracked <1 1 &ptetsize>
set &p to 1
set &r to 0
do while &r < &ptetrows
set &r up 1
log &vscale(,&r,1) &display(,&p,&ptetcols) &vscale(,&r,1)
set &p up &ptetcols
enddo
log '+' &hscale(,1,&ptetcols) '+'
log ' '
set &m to 0
do while &m < &messages
set &m up 1
log &mess(&m)
enddo
if &mismatch = 'Stop'
```

```
then
set &mess to 'Analyzer ended because Mismatch=' &mismatch
go to close
endif
if &mismatch = 'Pause'
then
compute &log-lines = &messages / remainder 8
if &log-lines > 0
then
do while &log-lines < 5
set &log-lines up 1
log ' '
enddo
endif
log 'REVIEW, then ROLL to visually compare full screen'
' displays.'
log 'STEP or VIEW to repeat next traced transaction.'
pause
endif
log ' '
endif
exit
*-------------------------------------------------------------------------*
* PAGE adds lines to separate message groups in log or monitor log. *
*-------------------------------------------------------------------------*
page
if &messages > 0
then
if &action = 'Pause'
then
compute &log-lines = &messages / remainder 8
if &log-lines > 0
then
do until &log-lines = 8
set &log-lines up 1
set &messages up 1
set &mess(&messages) to ' '
enddo
endif
else
if &mess(&messages) \= ' '
then
set &messages up 1
set &mess(&messages) to ' '
endif
endif
endif
exit
*-------------------------------------------------------------------------*
* LOAD processes analyzer option variables and trace list options. *
```

```
*--------------------------------------------------------------------*
load
set &dashes(,1,132) to all '='
set &hscale to '----+----1----+----2----+----3----+----4----+----5'
'----+----6----+----7----+----8----+----9----+----0'
'----+----1----+----2----+----3--'
set &vscale to '||||+||||1||||+||||2||||+||||3||||+||||4|||'
set &sessions to 0
set &compares to 0
set &rekeys to 0
set &mismatches to 0
set &keyerrors to 0
set &dsns to 0
set &dsni to 0
set &eof to 'no'
set &parm-length to length &trace
if &parm-length > 0
then
if &trace contains '('
then
compute &arg-i = &pteargi + 1
compute &arg-l = &pteargi - 1
do
if &arg-l > 0 and &trace(,1,&arg-l) > ''
then
set &dsn to &trace(,1,&arg-l)
if &trace(,&arg-i) contains ')'
then
compute &arg-l = &pteargi - &arg-i
if &arg-l > 0 and &trace(,&arg-i,&arg-l) > ''
then
set &mem to &trace(,&arg-i,&arg-l)
leave
endif
endif
endif
set &mess to 'TRACE=' &trace ' is invalid'
go to close
enddo
else
set &dsn to &trace
set &mem to &member
endif
if &mem > ''
then
set &dataset to &dsn '(' &mem ')'
set &dd to ''
allocate &dd dataset &dsn member &mem shr error *
else
set &dataset to &dsn
```

```
set &dd to ''
allocate &dd dataset &dsn shr error *
endif
if &pteresp \= '0000'
then
set &mess to 'Allocate failed for TRACE=' &dataset
go to close
endif
else
set &dataset to 'TRACE DD'
set &dd to 'TRACE'
endif
open input &dd unpack error *
if &pteresp \= '0000'
then
set &mess to ' open failed, rc=' &pteresp
go to close
endif
read &dd into &record endfile * error *
if &pteresp \= '0000'
then
if &pteresp \= '0804'
then
set &mess to ' is empty'
else
set &mess to ' read failed, rc=' &pteresp
endif
go to close
endif
set &lrecl to length &record
if &lrecl >= 80 and &lrecl <= 256
then
set &cc to capital &record(,1,1)
if &cc eq '*' or (&cc ge 'A' and &cc le 'Z')
then
log &dashes
log 'Loading Trace List Data: ' &dataset
log ' '
do until &pteresp \= '0000'
log ' ' &record(,1,80)
do
if &record(,1,1) = '*'
then
leave
endif
set &record to capital &record(,1,72)
if &record(,1,8) = 'ANALYZE='
then
set &dsns up 1
set &delays(&dsns) to &delays
```

```
set &delaypct(&dsns) to &delaypct
set &time(&dsns) to &time
set &synctime(&dsns) to &synctime
set &attr(&dsns) to &attr
set &image(&dsns) to &image
set &case(&dsns) to &case
set &logon(&dsns) to &logon
set &mask(&dsns) to &mask
set &type(&dsns) to &type
set &edit(&dsns) to &edit
set &filter(&dsns) to &filter
set &sync(&dsns) to &sync
set &logoff(&dsns) to &logoff
set &tracepds(&dsns) to &tracepds
set &screens(&dsns) to &screens
set &mismatch(&dsns) to &mismatch
set &keyerror(&dsns) to &keyerror
set &link(&dsns) to &link
set &link to ''
set &arg to &record(,9)
if &arg contains '('
then
compute &arg-l = &pteargi - 1
if &arg-l > 0
then
set &dsn(&dsns) to &arg(,1,&arg-l)
set &arg to &arg(,&pteargi)
if &arg contains ')' and &pteargi > 2
then
compute &arg-l = &pteargi - 2
set &mem(&dsns) to &arg(,2,&arg-l)
leave
endif
endif
else
if &arg contains ' ' and &pteargi > 1
then
compute &arg-l = &pteargi - 1
set &dsn(&dsns) to &arg(,1,&arg-l)
set &mem(&dsns) to ''
leave
endif
endif
set &mess to 'ANALYZE dataset is invalid'
go to close
endif
if &record(,1,7) = 'DELAYS='
then
set &delays to &record(,8,1)
leave
```

```
endif
if &record(,1,9) = 'DELAYPCT='
then
if &record(,10,8) contains ' '
then
compute &arg-l = &pteargi - 9
else
set &arg-l to 8
endif
if &arg-l > 0
then
set &delaypct to &record(,10,&arg-l)
else
set &delaypct to ''
endif
leave
endif
if &record(,1,5) = 'TIME='
then
set &arg-i to 6
do until &arg-i > 13 or &record(,&arg-i,1) < '0'
set &arg-i up 1
enddo
compute &arg-l = &arg-i - 6
if &arg-l > 0
then
set &time to &record(,6,&arg-l)
else
set &time to ''
endif
leave
endif
if &record(,1,9) = 'SYNCTIME='
then
set &arg-i to 10
do until &arg-i > 17 or &record(,&arg-i,1) < '0'
set &arg-i up 1
enddo
compute &arg-l = &arg-i - 10
if &arg-l > 0
then
set &synctime to &record(,10,&arg-l)
else
set &synctime to ''
endif
leave
endif
if &record(,1,5) = 'ATTR='
then
set &attr to &record(,6,1)
```

```
leave
endif
if &record(,1,6) = 'IMAGE='
then
set &image to &record(,7,1)
leave
endif
if &record(,1,5) = 'CASE='
then
set &case to &record(,6,1)
leave
endif
if &record(,1,6) = 'LOGON='
then
if &record(,7,8) contains ' '
then
compute &arg-l = &pteargi - 7
else
set &arg-l to 8
endif
if &arg-l > 0
then
set &logon to &record(,7,&arg-l)
else
set &logon to ''
endif
leave
endif
if &record(,1,5) = 'MASK='
then
if &record(,6,8) contains ' '
then
compute &arg-l = &pteargi - 6
else
set &arg-l to 8
endif
if &arg-l > 0
then
set &mask to &record(,6,&arg-l)
else
set &mask to ''
endif
leave
endif
if &record(,1,5) = 'LINK='
then
if &record(,6,8) contains ' '
then
compute &arg-l = &pteargi - 6
else
```

```
set &arg-l to 8
endif
if &arg-l > 0
then
set &link-l to length &link
set &link-l up 1
set &link(,&link-l,8) to &record(,6,&arg-l)
endif
leave
endif
if &record(,1,5) = 'TYPE='
then
if &record(,6,8) contains ' '
then
compute &arg-l = &pteargi - 6
else
set &arg-l to 8
endif
if &arg-l > 0
then
set &type to &record(,6,&arg-l)
else
set &type to ''
endif
leave
endif
if &record(,1,5) = 'EDIT='
then
if &record(,6,8) contains ' '
then
compute &arg-l = &pteargi - 6
else
set &arg-l to 8
endif
if &arg-l > 0
then
set &edit to &record(,6,&arg-l)
else
set &edit to ''
endif
leave
endif
if &record(,1,7) = 'FILTER='
then
if &record(,8,8) contains ' '
then
compute &arg-l = &pteargi - 8
else
set &arg-l to 8
endif
```

```
if &arg-l > 0
then
set &filter to &record(,8,&arg-l)
else
set &filter to ''
endif
leave
endif
if &record(,1,5) = 'SYNC='
then
if &record(,6,8) contains ' '
then
compute &arg-l = &pteargi - 6
else
set &arg-l to 8
endif
if &arg-l > 0
then
set &sync to &record(,6,&arg-l)
else
set &sync to ''
endif
leave
endif
if &record(,1,7) = 'LOGOFF='
then
if &record(,8,8) contains ' '
then
compute &arg-l = &pteargi - 8
else
set &arg-l to 8
endif
if &arg-l > 0
then
set &logoff to &record(,8,&arg-l)
else
set &logoff to ''
endif
leave
endif
if &record(,1,9) = 'TRACEPDS='
then
if &record(,10,44) contains ' '
then
compute &arg-l = &pteargi - 10
else
set &arg-l to 44
endif
if &arg-l > 0
then
```

```
set &tracepds to &record(,10,&arg-l)
else
set &tracepds to ''
endif
leave
endif
if &record(,1,8) = 'SCREENS='
then
set &screens to &record(,9,1)
leave
endif
if &record(,1,9) = 'MISMATCH='
then
set &mismatch to &record(,10,1)
leave
endif
if &record(,1,9) = 'KEYERROR='
then
set &keyerror to &record(,10,1)
leave
endif
set &mess to 'Trace list keyword is invalid.'
go to close
enddo
read &dd into &record endfile * error *
enddo
if &pteresp \= '0804'
then
set &mess to &dataset ' read failed, rc=' &pteresp
go to close
endif
close &dd
if &dd \= 'TRACE'
then
free &dd
endif
if &dsns = 0
then
set &mess to 'ANALYZE=dataset not found in ' &dataset
go to close
endif
set &dsni to 1
perform open
exit
else
if &lrecl >= 135
then
close &dd
perform open
exit
```

```
endif
endif
endif
set &mess to &dataset ' is not a trace or a trace list dataset'
go to close
*-------------------------------------------------------------------------*
* OPEN gets options and opens a session trace dataset to be analyzed. *
*-------------------------------------------------------------------------*
open
if &dsni > 0
then
set &dsn to &dsn(&dsni)
set &mem to &mem(&dsni)
set &delays to &delays(&dsni)
set &delaypct to &delaypct(&dsni)
set &time to &time(&dsni)
set &synctime to &synctime(&dsni)
set &attr to &attr(&dsni)
set &image to &image(&dsni)
set &case to &case(&dsni)
set &logon to &logon(&dsni)
set &mask to &mask(&dsni)
set &type to &type(&dsni)
set &edit to &edit(&dsni)
set &filter to &filter(&dsni)
set &sync to &sync(&dsni)
set &logoff to &logoff(&dsni)
set &tracepds to &tracepds(&dsni)
set &screens to &screens(&dsni)
set &mismatch to &mismatch(&dsni)
set &keyerror to &keyerror(&dsni)
set &link to &link(&dsni)
if &mem > ''
then
set &dataset to &dsn '(' &mem ')'
else
set &dataset to &dsn
endif
endif
set &delaypct from string &delaypct
set &time from string &time
set &synctime from string &synctime
do
if &delays(,1,1) contains 's'
then
set &delays to 'Synctime'
leave
endif
if &delays(,1,1) contains 't'
then
```

```
set &delays to 'Time'
leave
endif
if &delays(,1,1) contains 'b'
then
set &delays to 'Bracket'
leave
endif
if &delays(,1,1) contains 'i'
then
set &delays to 'Input'
leave
endif
if &delaypct \= ''
then
set &delays to 'Bracket'
leave
endif
if &time \= ''
then
set &delays to 'Time'
leave
endif
set &delays to 'Synctime'
enddo
if &delays = 'Input'
then
delay input
else
delay bracket
endif
if &delaypct = '' or &delaypct is not decimal
then
set &delaypct to '100'
endif
delay percent &delaypct
if &time = '' or &time is not decimal
then
set &time to '1000'
endif
if &synctime = '' or &synctime is not decimal
then
set &synctime to '1000'
endif
if &attr(,1,1) contains 'n' /* attribute compare */
then /* default=yes */
set &attr to 'No'
else
set &attr to 'Yes'
endif
```

```
if &image(,1,1) contains 'y' /* compare image */
then /* default=no */
set &image to 'Yes'
else
set &image to 'No'
endif
if &case(,1,1) contains 'n' /* compare case */
then /* default=yes */
set &case to 'No'
else
set &case to 'Yes'
endif
if &screens(,1,1) contains 'c' /* screens=chains|freekb */
then
set &screens to 'Chains'
else
set &screens to 'Freekb'
endif
if &ptepgm = 'PTEIS'
then
if &mismatch(,1,1) contains 'c'
then
set &mismatch to 'Continue'
else
set &mismatch to 'Pause'
endif
if &keyerror(,1,1) contains 'c'
then
set &keyerror to 'Continue'
else
set &keyerror to 'Pause'
endif
else
if &mismatch(,1,1) contains 's'
then
set &mismatch to 'Stop'
else
set &mismatch to 'Continue'
endif
if &keyerror(,1,1) contains 'c'
or &keyerror(,1,1) contains 'p'
then
set &keyerror to 'Continue'
else
set &keyerror to 'Stop'
endif
endif
log &dashes
log 'Active Analyzer Options:'
log ' '
```

```
log ' *'
log ' * Timing Options:'
log ' *'
log ' DELAYS=' &delays
log ' DELAYPCT=' &delaypct
log ' TIME=' &time
log ' SYNCTIME=' &synctime
log ' *'
log ' * Compare Options:'
log ' *'
log ' IMAGE=' &image
log ' CASE=' &case
log ' ATTR=' &attr
log ' *'
log ' * Scripts:'
log ' *'
log ' LOGON=' &logon
log ' MASK=' &mask
log ' TYPE=' &type
log ' EDIT=' &edit
log ' FILTER=' &filter
log ' SYNC=' &sync
log ' LOGOFF=' &logoff
log ' *'
log ' * Trace Upgrade Options:'
log ' *'
log ' TRACEPDS=' &tracepds
log ' SCREENS=' &screens
log ' *'
log ' * Event Handling Options:'
log ' *'
log ' MISMATCH=' &mismatch
log ' KEYERROR=' &keyerror
log ' '
log 'Now Processing:'
log ' '
do while &link > ''
log ' LINK=' &link(,1,8)
link &link(,1,8)
set &link to &link(,9)
enddo
log ' ANALYZE=' &dataset
log ' '
if &dsni > 0
then
set &dd to ''
if &mem > ''
then
allocate &dd dataset &dsn member &mem shr error *
else
```

```
allocate &dd dataset &dsn shr error *
endif
if &pteresp \= '0000'
then
set &mess to &dataset ' allocate failed rc=' &pteresp
go to close
endif
endif
open input &dd error *
if &pteresp \= '0000'
then
set &mess to &dataset ' open failed, rc=' &pteresp
go to close
endif
do
if &ptestate = 'OFFLINE'
then
if &logon = ''
then
set &auto-logon to 'ok'
leave
endif
log 'Linking to logon script: ' &logon
link &logon
if &ptestate = 'OFFLINE'
then
set &mess to 'Logon failed. Processing ended.'
go to close
endif
set &sessions up 1
endif
set &auto-logon to 'no'
enddo
set &t-responses to 0
set &s-responses to 0
if &tracepds > '' and &mem > ''
then
log 'Tracing new session to ' &tracepds '(' &mem ')'
log ' '
set &trace-dd to ''
allocate &trace-dd dataset &tracepds member &mem shr error *
if &pteresp \= '0000'
then
set &mess to 'TRACEPDS member allocate failed rc=' &pteresp
go to close
endif
if &screens = 'Chains'
then
trace to &trace-dd format pack logons chains
else
```

```
trace to &trace-dd format pack logons freekb
endif
endif
exit
*-------------------------------------------------------------------------*
* TRACK tracks next screen/statement and manages trace list datasets. *
*-------------------------------------------------------------------------*
track
do
if &eof \= 'yes'
then
if &delays = 'Synctime'
then
track &dd logical statement endfile * error *
else
track &dd transaction statement endfile * error *
endif
if &pteresp = '0000' or &pteresp = 'LINE'
then
leave
endif
if &pteresp \= '0804'
then
set &mess to &dataset ' track error rc=' &pteresp
go to close
endif
if &ptestate \= 'OFFLINE'
then
invite 500
endif
close &dd
if &dd ne 'TRACE'
then
free &dd
endif
if &trace-dd > ''
then
endtrace
free &trace-dd
set &trace-dd to ''
endif
if &ptestate ne 'OFFLINE'
link &sync
if &dsni < &dsns
then
set &dsni up 1
perform open
iterate
endif
if &compares > 0
```

```
then
set &mess to 'All traced transactions were repeated'
' and responses were compared.'
else
set &mess to ' '
endif
go to close
endif
enddo
exit
*-----------------------------------------------------------------------*
* CLOSE signals EOF at end of data or at an unresolvable error. *
*-----------------------------------------------------------------------*
close
set &eof to 'yes'
exit
*-----------------------------------------------------------------------*
* RETURN reports final stats, performs cleanup, then exits or ends. *
*-----------------------------------------------------------------------*
return
if &ptestate \= 'OFFLINE'
then
invite 500
if &ptestate \= 'OFFLINE'
then
link &logoff
endif
endif
compute &ss(,1,5) = &sessions justify right
compute &rs(,1,5) = &rekeys justify right
compute &cs(,1,5) = &compares justify right
compute &ms(,1,5) = &mismatches justify right
compute &ks(,1,5) = &keyerrors justify right
log &dashes log ' '
log &mess
log &ss ' new sessions started.'
log &rs ' old input screens were rekeyed as new transaction input.'
log &ks ' old input screens contained data that could not be typed.'
log &cs ' new transaction responses were compared to old responses.'
log &ms ' new response screens did not match old response screens.'
if &ptepgm = 'PTEIS'
then
log 'STEP again to end test.'
pause
endif
log ' ' log &dashes
```

Batch Multi-Port Applications

Because CA Enterprise Tester scripts can use online systems in the same way that people, batch jobs may be used to run CA Enterprise Tester multi-port applications. For example, unattended multi-session tests may be performed at night by scheduling a job created by the CA Enterprise Tester Multi-Port Watch Utility. A job created by the Multi-Port Watch Utility uses a distributed CA Enterprise Tester script, PTEMPLAY to start and watch a multi-session test using TSO and the CA Enterprise Tester Interactive System in the same way that a person would.

The following script is the distributed version of PTEMPLAY. For a description of the functions performed by PTEMPLAY, refer to The Multi-Port Watch Utility.

PTEMPLAY

```
*-------------------------------------------------------------------*
* PTEMPLAY - The Multi-Port Play Manager used by the Watch Utility. *
*-------------------------------------------------------------------*
* PTEMPLAY uses a TSO session and the ProTerm Interactive System to *
* play a select script and to watch script activity. The ProTerm *
* Multi-Port Watch Utility creates jobs that use PTEMPLAY to manage *
* unattended multi-session online system tests. *
*-------------------------------------------------------------------*
* Multi-Port Play Batch Job Example: *
* *
* //jobname JOB ... *
* //PTEMPLAY EXEC PGM=PTEBS *
* //PTELIB DD DISP=SHR,DSN=uid.proterm.script *
* //PTELOG DD SYSOUT=A *
* //PTEIN DD * *
* set &library to "'uid.proterm.script'" *
* set &logon to 'tsologon' *
* set &logoff to '' *
* set &select to 'mpselect' *
* set &ports to 50 *
* set &delay to 100 *
* set &parms to 'rate=25,ports=50' *
* set &logons to 0 *
* set &runtime to '0100' *
* set &review to 4 *
* set &logdsn to "'uid.proterm.log'" *
* set &logmem to 'multilog' *
* link 'PTEMPLAY' *
* /* *
* *
*-------------------------------------------------------------------*
* Start run clock and calculate expiration time.
time
```

```
set &start-time to &ptetime(,1,14)
if &runtime is decimal and &runtime > 0
then
set &stop-time to &start-time
set &runtime(,1,4) to zoned &runtime
compute &minute = &start-time(,11,2) + &runtime(,3,2)
if &minute > 59
then
compute &hour = &minute / quotient 60
compute &minute = &minute - &hour * 60
endif
set &stop-time(,11,2) to zoned &minute
compute &hour = &hour + &start-time(,9,2) + &runtime(,1,2)
if &hour > 23
then
compute &day = &hour / quotient 24
compute &hour = &hour - &day * 24
endif
set &stop-time(,9,2) to zoned &hour
if &day > 0
then
set &date to '&start-time(,1,8) "yyyymmdd" plus &day '
'format &stop-time(,1,8) "yyyymmdd"'
link 'ptedate'
endif
log 'Multi-Play Job Stop clock set to '
&stop-time(,1,4) '/' &stop-time(, 5,2) '/' &stop-time(, 7,2) ' '
&stop-time(,9,2) ':' &stop-time(,11,2) ':' &stop-time(,13,2)
else
set &stop-time to '99999999999999999999'
endif
do
if &ports is not decimal or &ports < 1 or &ports > 999
then
log 'ports=' &ports ' is not 1-999'
leave
endif
* Link to user logon script to start TSO/ISPF/ProTerm session.
link &logon
if &ptestate = 'OFFLINE' or <1 2 *> \= 'ProTerm Control'
then
set &mess to 'ProTerm session not started by ' &logon
leave
endif
* Save and set ProTerm Options
log 'Setting ProTerm Options'
type home 'options' enter /* enter options command*/
if <1 2 *> \= 'ProTerm Options'
then
set &mess to 'Options Panel page 1 not Displayed'
```

```
leave
endif
if &review is not decimal or &review < 0
then
set &review to 0
endif
set &options-panel-1 to image <1 1 &ptesize>
type home 'down'
tab /* scroll ==> */
tab 'yes' /* set session to zoom */
tab 'no ' /* end all on return ? */
tab /* dsn prefix type */
tab erase &review /* screen review limit */
tab '0020' /* monitor log lines */
tab /* invite interval */
tab '0000' /* step time for movies */
tab 'no ' /* play zoom mode */
enter
if <1 2 *> \= 'ProTerm Options'
then
set &mess to 'Options Panel page 2 not displayed'
leave
endif
set &options-page-2 to image <1 1 &ptesize>
type home 'end'
tab /* scroll ==> */
tab /* ispf stats */
tab /* protect dataset name */
tab 'yes' /* think/wait delay */
tab 'yes' /* pa2 reshow */
tab 'no ' /* select exits */
tab 'yes' /* show free kb screens */
enter
* Start Multi-Port Script Log
if &logdsn > ''
then
if &logmem = ''
then
log 'Starting Multi-Port Log to ' &logdsn
else
log 'Starting Multi-Port Log to ' &logdsn '('&logmem ')'
endif
type 'log' enter
if <1 2 *> \= 'ProTerm Log'
then
set &mess to 'ProTerm Log Panel not displayed'
leave
endif
set &log-panel to image <1 1 &ptesize>
type home tab erase &logdsn /* dsn/sysout */
```

```
tab erase &logmem /* mem/blank */
tab 'no' /* no ru log */
tab 'no' /* replace log data*/
tab 'yes' /* share log pds */
enter
if <1 2 *> \= 'ProTerm Control'
then
set &mess to 'Log was not successfully started'
leave
endif
endif
* Play script on multiple ports
log 'Playing ' &select ' on ' &ports ' ports'
if &delay = ' '
then
set &delay to '100'
endif
if &think = ' '
then
set &think to '0000'
endif
if &logons is decimal and &logons > 0 and &logons < &ports
then
set &multiple to &logons
set &command to 'play'
else
set &multiple to &ports
set &command to 'monitor'
endif
set &played to 0
do while &played < &ports /* request multi-play */
compute &port = &played + 1
type home erase 'play ' &port enter
if <1 2 *> \= 'ProTerm Play'
then
set &mess to 'ProTerm Play Panel not displayed'
leave
endif
if &play-panel = '' /* save panel data */
then
set &play-panel to image <1 1 &ptesize>
endif
set &played up &multiple
if &played > &ports
then
compute &multiple = &multiple + &ports - &played
set &played to &ports
endif
type home &command
tab erase &library
```

```
tab erase &select
tab erase &parms
tab tab tab
tab erase &delay
tab erase &think
tab erase &multiple
enter
if <1 1 &ptesize> contains 'Press ENTER to confirm request'
then
type enter /* confirm multi-play */
endif
if <1 2 *> \= 'ProTerm Control'
then
set &mess to 'Multi-port play request failed'
leave
endif
if &command = 'play' /* pace play logons */
and &played < &ports
then
do until &port > &played
compute &portf(,1,3) = &port justify right
set &portlist to &portlist &portf
set &port up 1
enddo
set &multiple to 0
do until &multiple > 0
delay 100
type enter
set &portl to length &portlist
compute &portp = &portl + 1
do while &portp > 3
set &portp down 3
if &portlist(,&portp,3) < <11,3,3>
or (<22,3,3> > ' '
and &portlist(,&portp,3) > <22,3,3>)
then
compute &port = &portlist(,&portp,3) - 11
if &port < 1
then
set &port to 1
endif
type home erase 'locate ' &port enter
endif
compute &row =
11 + &portlist(,&portp,3) - <11,3,3>
set &row to capital <&row 1 80>
if &row(,34,3) = 'I/O'
or &row(,34,2) = 'IN'
or &row(,34,3) = 'OUT'
or &row(,46,3) \= '#EX'
```

```
then
set &multiple up 1
compute &portn = &portp + 3
set &portlist(,&portp) to &portlist(,&portn)
set &portl down 3
if &portl > 0
then
set &portlist to &portlist(,1,&portl)
else
set &portlist to ''
endif
endif
enddo
enddo
endif
enddo
if &mess > ' '
then
leave
endif
if &command = 'monitor'
then
type home erase 'resume' enter
endif
* Monitor played scripts until 000 Active Ports
log 'Now Monitoring ' &ports ' played scripts'
do until <1,1,&ptecols> contains '000 active ports'
delay 3000
type home 'locate 1' enter
set &port to 1
do until &port > &ports
if <11 1 960> contains 'at error'
then
set &ports-at-error up 1
set &row to image <&pteargr 1,80>
log 'Port' &row
compute &port# = <&pteargr,2,4>
do
type home 'monitor ' &port# enter
if <1 2 *> \= 'ProTerm Monitor'
then
set &mess to 'Monitor Panel not displayed'
leave
endif
dump screen
if &review < 1
then
leave
endif
compute &screens = &review + 1
```

```
type home 'review ' &screens enter
do while <1 2 *> = 'ProTerm Session Review'
dump screen
if <1,24,4> = '(-1,'
then
leave
endif
type home 'step' enter
enddo
enddo
if &mess > ' '
then
leave
endif
set &ends to 0
do until <1 2 *> = 'ProTerm Control Panel'
if &ends > 2
then
set &mess to 'Unable to return from Review'
leave
endif
type home 'end' enter
set &ends up 1
enddo
if &mess > ' '
then
leave
endif
type home 'end ' &port# enter
iterate
endif
if <11 1 960> contains 'stopped'
then
set &port# to <&pteargr 2 3>
type home 'resume ' &port# enter
iterate
endif
set &port up 12
if &port <= &ports
then
type home 'locate ' &port enter
endif
enddo
time
if &ptetime gt &stop-time
then
set &mess to 'Stopping scripts at run time limit'
leave
endif
enddo
```

```
enddo
* Document any problem, end any active tasks, end ProTerm session
if &mess > ' '
then
log 'Error: ' &mess
log 'Multi-Port Play Terminated'
dump screen
else
log 'All scripts now ended'
endif
set &ends to 0
do until &ptestate = 'OFFLINE'
or <1 2 *> = 'ProTerm Control'
or &ends > 3
type home erase 'end' enter
set &ends up 1
enddo
if &ptestate \= 'OFFLINE'
and <1 2 *> = 'ProTerm Control'
then
set &ports-active to <1 65 3>
type home 'stop' enter
type home 'end play' enter
type home 'end session' enter
type home 'end log' enter
if &options-page-2 > ''
then
type home erase 'options' enter
type overlay <1 1> &options-panel-1
type home erase 'down' enter
type overlay <1 1> &options-page-2
type home erase 'end' enter
endif
if &log-panel > ''
then
type home erase 'log' enter
type overlay <1 1> &log-panel
type home erase 'end' enter
endif
if &play-panel > ''
then
type home erase 'play' enter
type overlay <1 1> &play-panel
type home erase 'end' enter
endif
endif
log 'Ending TSO/ProTerm session'
link &logoff
if &ptestate \= 'OFFLINE'
then
```

```
type home erase '=x' enter
type 'logoff' enter
endif
if &ptestate \= 'OFFLINE'
then
dump screen
logoff
endif
* Log parameters and results
log 'Multi-Play Parms and Results:'
log '+------------------------------------------------------------'
log '| Script library = ' &library
log '| Select script = ' &select
log '| Logon script = ' &logon
log '| Logoff script = ' &logoff
log '| Play on ports = ' &ports
log '| Delay Percent = ' &delay
log '| Think Time = ' &think
log '| Script Parms = ' &parms
log '| Maximum Logons = ' &logons
log '| Run Time Limit = ' &runtime
log '| Review Screens = ' &review
log '| Log Data Set = ' &logdsn
log '| Log Member = ' &logmem
log '|'
log '| Incomplete ports : ' &ports-active
log '| Ports at error : ' &ports-at-error
log '|'
log '| Started : '
&start-time(,1,4) '/' &start-time(,5,2) '/' &start-time(,7,2) ' '
&start-time(,9,2) ':' &start-time(,11,2) ':' &start-time(,13,2)
time
log '| Ended : '
&ptetime(,1,4) '/' &ptetime(,5,2) '/' &ptetime(,7,2) ' '
&ptetime(,9,2) ':' &ptetime(,11,2) ':' &ptetime(,13,2)
set &date to '&start-time(,1,8) "yyyymmdd" format &start-day "day"'
link 'ptedate'
set &date to '&ptetime(,1,8) "yyyymmdd" format &end-day "day"'
link 'ptedate'
compute &runtime = 86400 * (&end-day - &start-day)
+ 3600 * (&ptetime(, 9,2) - &start-time(, 9,2))
+ 60 * (&ptetime(,11,2) - &start-time(,11,2))
+ &ptetime(,13,2) - &start-time(,13,2)
compute &days = &runtime / quotient 86400
compute &runtime = &runtime - 86400 * &days
compute &hours = &runtime / quotient 3600
compute &runtime = &runtime - 3600 * &hours
compute &minutes = &runtime / quotient 60
compute &seconds = &runtime - 60 * &minutes
log '| Elapsed Time : ' &days ' days, '
```

```
&hours ' hours, '
&minutes ' minutes and '
&seconds ' seconds'
log '+------------------------------------------------------------'
```

A Central Encoded Password Maintenance System

This sample provides a system that may be used to maintain a central file of encoded user passwords in a VSAM KSDS, and a script subroutine that may be used in user-written scripts to access current encoded passwords. The system allows scripts to be written that do not contain encoded user passwords. Scripts that use the script subroutine described here, require no maintenance when user passwords are changed.

The system uses a VSAM KSDS to maintain a file of encoded user passwords. USERIDis used as a record key. Each record contains two fields of data; (1) the length of an encoded user password and (2) an encoded user password. The following job, member PTECODVS in the distributed CA Enterprise Tester JCL library, may be used to define and initialize the VSAM encoded password file.

```
//*-------------------------------------------------------------------*
//* THIS JOB MAY BE USED TO DEFINE AND INITIALIZE A VSAM *
//* FILE THAT IS MAINTAINED AND USED BY CLIST, PTECODE, AND *
//* PROTERM SCRIPTS, PTECODEU AND PTECODE. *
//*-------------------------------------------------------------------*
//DEFINE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DEL PTE.CODE.FILE
DEF CL -
(NAME(PTE.CODE.FILE),VOL(??????),SPEED,TRK(1,1) -
RECSZ(18,18),KEYS(8,0),SHR(2,3),IMBD,REPL) -
DATA(NAME(PTE.CODE.FILE.DATA),CISZ(4096)) -
INDEX(NAME(PTE.CODE.FILE.INDEX),CISZ(2048))
//LOAD EXEC PGM=PTEBS
//STEPLIB DD DSN=PTE.R310.LOADLIB,DISP=SHR
//PTELOG DD SYSOUT=A
//PTEIN DD *
ALLOCATE CODES DATASET 'PTE.CODE.FILE' OLD
OPEN CODES OUTPUT
SET &RECORD(,1,18) TO ALL ' '
SET &RECORD(,9,2) TO BINARY 0
WRITE CODES FROM &RECORD
CLOSE CODES
FREE CODES
```

The system allows system users to update a record in the VSAM encoded password file whenever they change their password. A CLIST is used to display an ISPF panel that contains a non-display input field. When the user changes their password, they may use the CLISTto update their record in the VSAM encoded password file. The CLISTis provided in member PTECODEof the distributed CA Enterprise Tester EXEC library.

```
/* THIS CLIST, PTECODE, DISPLAYS AN ISPF PANEL, THAT CONTAINS */
/* A NON-DISPLAY INPUT FIELD. WHEN A USER CHANGES THEIR */
/* PASSWORD, THEY MAY USE THE CLIST TO UPDATE THEIR RECORD IN */
/* A VSAM ENCODED PASSWORD FILE. THE CLIST CALLS A PROTERM */
/* SCRIPT TO ENCODE THE PASSWORD AND UPDATE THE VSAM FILE. */
SET &MSG =
SET &LOOP = YES
DO WHILE &LOOP = YES
SET &CODE =
ISPEXEC DISPLAY PANEL(PTECODEP) MSG(&MSG)
SET &CC = &LASTCC
IF &CC ª= 0 -
THEN DO
EXIT
END
ALLOC F(PTEIN) DA('PTE.R310.SCRIPT(PTECODEU)') SHR
ALLOC F(PTELOG) DUMMY
CALL 'PTE.R310.LOADLIB(PTEBS)' 'CODE=&CODE'
SET &CC = &LASTCC
FREE F(PTEIN)
FREE F(PTELOG)
IF &CC = 0 THEN SET &MSG = PTEC010
IF &CC = 1 THEN SET &MSG = PTEC011
IF &CC = 2 THEN SET &MSG = PTEC012
IF &CC = 3 THEN SET &MSG = PTEC013
END
```

The ISPF panel displayed by the PTECODE CLIST, is provided in member PTECODEP of the distributed CA Enterprise Tester PLIB library, and is shown below.

```
)ATTR
ª TYPE(INPUT) INTENS(NON)
)BODY
%-------------------------- ProTerm Password Update ----------------------------
%COMMAND ===>_ZHCMD
+
%Enter Password to be encoded and stored:
+Password%===>ªCODE +
)INIT
.CURSOR = CODE
)PROC
VER(&CODE,NONBLANK)
)END
```

The PTECODE CLIST also displays messages that tell the user the results of an attempt to update their encoded password record. The messages, provided in member PTEC01 of the distributed CA Enterprise Tester MLIB library, are shown below.

```
PTEC010 'Password updated'
'The password entered was encoded and stored.'
PTEC011 'Password invalid' .ALARM=YES
'The password entered was invalid and could not be encoded.'
PTEC012 'VSAM alloc failed' .ALARM=YES
'The VSAM encoded password file could not be allocated.'
PTEC013 'VSAM update failed' .ALARM=YES
'An add or update to the VSAM encoded password file failed.'
```

The PTECODE CLIST calls the ProTerm Script Language processor, program PTEBS, to execute a ProTerm script to encode a user password and to update the user's record in the VSAM encoded password file. The script uses the subroutine documented in Token Security Subroutine on page 4-9 to encode the password. VSAM allocation techniques used in the script and the access script that follows insures full integrity VSAM sharing. The encoded password maintenance script, provided in member PTECODEU of the distributed ProTerm script library, is shown below.

```
*-------------------------------------------------------------------** PTECODEU
- Update an encoded password in a VSAM password code file, **--------------------
-----------------------------------------------** This script accepts a
password that is provided as a parameter,   ** encodes the password for use by
the user who is executing the     ** script, and updates the user's record in a
VSAM encoded password  ** file.     **-----------------------------------------
-------------------------*
    set &user(,1,8)  to &pteuser    set &code-len       to length &code    set
&parm-len(,1,2) to binary &code-len    set &code(,1,8)  to &code    set &encode-
parm  to 'ENCODE  ' &user &parm-len &code    call pteis using &encode-parm    if
&pterc > 0    then
  end 1
    else  set &allocated to 'no'  set &retries to 0  do until &allocated = 'yes'
allocate codefile dataset 'pte.codefile' old error *if &pteresp = '0000'then
    set &allocated to 'yes'
else    set &retries up 1    if &retries < 100    then
  wait 100    else end 2    endif
endif  enddo  open codefile output  read codefile key &user into &record notfound
*  if &pteresp = '0000' then
update codefile from &user &encode-parm(,17,10) error *
 else
write  codefile from &user &encode-parm(,17,10) error *  endif  if &pteresp =
'0000' then
set &rc to 0
 else
set &rc to 3  endif  close codefile free  codefile
    endif    end &rc
```

The final item in the encoded password maintenance system is a ProTerm script subroutine that may be used by other scripts to access the VSAM encoded password file and to get the current encoded password of the user who is executing the script. The encoded password access routine, provided in member PTECODE of the distributed CA Enterprise Tester script library, is shown below.

```
*-------------------------------------------------------------------*
* PTECODEU - Update an encoded password in a VSAM password code file, *
*-------------------------------------------------------------------*
* This script accepts a password that is provided as a parameter, *
* encodes the password for use by the user who is executing the *
* script, and updates the user's record in a VSAM encoded password *
* file. *
*-------------------------------------------------------------------*
set &user(,1,8) to &pteuser
set &code-len to length &code
set &parm-len(,1,2) to binary &code-len
set &code(,1,8) to &code
set &encode-parm to 'ENCODE ' &user &parm-len &code
call pteis using &encode-parm
if &pterc > 0
then
end 1
else
set &allocated to 'no'
set &retries to 0
do until &allocated = 'yes'
allocate codefile dataset 'pte.codefile' old error *
if &pteresp = '0000'
then
set &allocated to 'yes'
else
set &retries up 1
if &retries < 100
then
wait 100
else
end 2
endif
endif
enddo
open codefile output
read codefile key &user into &record notfound *
if &pteresp = '0000'
then
update codefile from &user &encode-parm(,17,10) error *
else
write codefile from &user &encode-parm(,17,10) error *
endif
if &pteresp = '0000'
```

```
then
set &rc to 0
else
set &rc to 3
endif
close codefile
free codefile
endif
end &rc
```

Appendix B: Messages

This appendix documents CA Enterprise Tester system messages.

Messages

PTE00001 - PROTERM OPTIONS TABLE, PTEOT, IS NOT Rnnn

The CA Enterprise Tester options table loaded was not the correct release for the release of CA Enterprise Tester being executed. When the message is issued, Rnnn is replaced by the release number of the CA Enterprise Tester code executing when the invalid CA Enterprise Tester options table was loaded. To correct the problem, assemble the options table for release Rnnn and/or insure that the correct options table has been linked, copied or moved to the load library being used to access CA Enterprise Tester Rnnn modules.

PTE00100 - SIGNON USER ID NOT 1-8 NATIONAL CHARS

While using CA Enterprise Tester security options designed for use in systems not supported by an z/OS SAF based security system, security initialization failed because a user ID specified via a CA Enterprise Tester Interactive System security panel or other authorized user ID source was not 1 through 8 national characters (A-Z, 0-9, @, #or $).

PTE00101 - USER ID MAY NOT BE CHANGED WHILE SECURED TASKS ARE ACTIVE

While using CA Enterprise Tester security options designed for use in systems not supported by an z/OS SAF based security system, a request to change the active user ID failed because the user ID that was active when the change was requested was being used by active CA Enterprise Tester tasks.

PTE00102 - CAPTURE SECURITY NOT CHANGED BECAUSE SECURED TASKS ARE ACTIVE

A request to change the active password capture format to another format allowed by CA Enterprise Tester system options was rejected because CA Enterprise Tester tasks were using the active capture format to secure data when the request to change the active capture format was processed. The active o change the active password capture format may be changed to another allowed format only when no CA Enterprise Tester task is active that is using the active capture format to secure data.

PTE00103 - SPECIFIED CAPTURE OPTION IS NOT ALLOWED AND WAS IGNORED

A request to change the active password capture format to another format was rejected because the format specified was not allowed by CA Enterprise Tester system options. Password capture formats permitted by CA Enterprise Tester system options may be displayed by entering the SECURE command at the CA Enterprise Tester Interactive System Control Panel.

PTE00104 - AN INVALID CAPTURE FORMAT OPTION WAS IGNORED

A request to change the active password capture format to another format was rejected because the format specified was not a password capture format supported by CA Enterprise Tester. Password capture formats permitted by CA Enterprise Tester system options may be displayed by entering the SECURE command at the CA Enterprise Tester Interactive System Control Panel.

PTE00105 - TOKEN SECURITY INACTIVE BECAUSE NO USER ID AVAILABLE

CA Enterprise Tester Token security initialization failed because no valid user ID was available. If this message is issued in an z/OS system that is supported by a SAF-based security system, report the security system error to administrators or technicians responsible for your system security. If the problem persists, contact CA Enterprise Tester technical support for assistance in diagnosing the cause of the problem.

PTE00106 - ENCODE FOR USER ID IGNORE BECAUSE CODE FOR NOT ALLOWED

A request to encode captured password for a specified user ID was ignored either because CA Enterprise Tester system options do not permit passwords to be encoded for another user ID or because the authority was disabled per system options after an excessive number of passwords were encoded for another user ID.

PTE00107 - CODEFOR USER ID IGNORED BECAUSE USER ID NOT 2-8 CHARS

A request to encode captured password data using a specified user ID was rejected because the user ID specified was not 2 through 8 national characters (A-Z, 0-9, @, #or $).

PTE00108 - CODE SECURITY INACTIVE BECAUSE USER ID NOT 2-8 CHARS

CA Enterprise Tester Code security initialization failed because the active user ID was not 2 through 8 national characters (A-Z, 0-9, @, #or $). If your user ID is 2 through 8 national characters and this message is issued in an z/OS system that is supported by a SAF-based security system, report the security system error to administrators or technicians responsible for your system security. If the problem persists, and your user ID is 2 through 8 national characters, contact CA Enterprise Tester technical support for assistance in diagnosing the cause of the problem.

PTE00109 - SCREEN TRACE FIELD SECURITY FOOTNOTE IS INVALID

A traced screen image read by the PSL TRACK statement or Interactive System Browse Panel contained a 'TOKEN AT' footnote which are created by CA Enterprise Tester to identify traced screen fields that are secured using Token Security. However the footnote was invalid because it did not identify a valid non-display field using syntax used by CA Enterprise Tester to identify traced token fields. If CA Enterprise Tester created the footnote, report the error to CA Enterprise Tester technical support.

PTE00110 - THE TOKEN NAME SPECIFIED CONTAINED INVALID CHARACTERS

A token name specified at the CA Enterprise Tester Token Security was not a valid token name. A token name must be a valid PDS member name consisting of 1 through 8 national characters (A-Z, 0-9, @, # or $), but may not begin with a character, than 0 through 9.

PTE00111 - THE LIBRARY MEMBER SPECIFIED IS NOT A TOKEN SCREEN

The data in a library member specified at the CA Enterprise Tester Token Security Panel or selected from a library member list does not begin with a valid token screen. A valid token screen is a screen image formatted like a screen image captured using the CA Enterprise Tester trace FORMAT option, whose cursor is positioned at the first position of a modifiable non-display field that is large enough to contain the library member name which is used by CA Enterprise Tester as a password token.

PTE00112 - ENTER NON-NULL DATA TO UPDATE TOKEN

A request to update the value of a token was ignored because no token data was entered in the token field when the token screen was displayed. To update the value of a token or to create a new token enter the data to be represented by the token as instructed when a token screen is displayed.

PTE00113 - TYPE CODE IS NOT 3-8 NATIONAL CHARACTERS

A PSL TYPE statement or command for REXX specified the CODE keyword to enter an encoded password, however the value found following the CODE keyword was not 3 through 8 national characters (A-Z, 0-9, @, # or $). Specify a valid encoded password value when the TYPE statement or command CODE keyword is used to enter the password for an encoded value.

PTE00114 - TYPE TOKEN NAME IS NOT 1-8 NATIONAL CHARACTERS

A PSL TYPE statement or command for REXX specified the TOKEN keyword to enter a token value, however the data found following the TOKEN keyword was not a 1 through 8 national character token name (A-Z, 0-9, @, # or $). Specify the name of an existing password token when the TYPE statement or command TOKEN keyword is used to enter data represented by a CA Enterprise Tester password token.

PTE00115 - TYPE CODE CURSOR LOC VISABLE OR FIELD TOO SMALL

A PSL TYPE statement or command for REXX that specified the CODE keyword could not be processed because the cursor was positioned at a visible screen location, or because the length of the encoded data exceeded length of the non-display field into which the data was to be typed.

PTE00116 - TYPE TOKEN CURSOR LOC VISIBLE OR FIELD TOO SMALL

A PSL TYPE statement or command for REXX that specified the TOKEN keyword could not be processed because the cursor was positioned at a visible screen location, or because the token represents data that was previously entered into a non-display screen field that was larger than the field into which the token data was to be typed.

PTE00117 - TYPE TOKEN NOT FOUND IN USER PROFILE

A PSL TYPE statement or command for REXX that specified the TOKEN keyword could not be processed because a token value did not exist in the user's CA Enterprise Tester PROFILE data set. A token can only be used to TYPE confidential data when the token user has previously entered the confidential data to be typed when the token is used.

PTE00118 - TOKEN NOT TYPABLE BECAUSE NO USER ID AVAILABLE

A PSL TYPE statement or command for REXX that specified the TOKEN keyword could not be processed because the TYPE statement or command was executed in an address space for which no owning user ID had been established. In a system that is managed by a security system that identifies the owner of an address space as prescribed by z/OS SAF, the failure represents a probable security system failure. Either report the problem to administrators or technicians responsible for your system security or contact CA Enterprise Tester technical support for assistance in diagnosing the problem.

PTE00119 - CODE NOT TYPABLE BECAUSE NO USER ID OR USER ID LT 2 CHARS

A PSL TYPE statement or command for REXX that specified the CODE keyword could not be processed because the TYPE statement or command was executed in an address space for which no owning user ID had been established or the owning user ID was not 2 through 8 national characters (A-Z, 0-9, @, # or $). If a valid user ID should have been available, the failure represents a probable security system failure. Either report the problem to administrators or technicians responsible for your system security or contact CA Enterprise Tester technical support for assistance in diagnosing the problem.

PTE00120 - TOKEN NOT TYPED BECAUSE TOKEN DATA NOT DECODABLE FOR USER ID

A PSL TYPE statement or command for REXX that specified the TOKEN keyword could not be processed because the user ID that owned the address space in which the TYPE statement or command was executed was not a valid key for the encrypted token data. Possible causes of the error are (1) a corrupted CA Enterprise Tester PROFILE data set, (2) PROFILE data was copied from another user's PROFILE data set, or (3) the correct user ID was not available due to a security system error. Either report the problem to administrators or technicians responsible for your system security or contact CA Enterprise Tester technical support for assistance in diagnosing the problem.

PTE00121 - THE CODE FOR THE ENTERED DATA AND USER ID IS "code")

A request to create a CA Enterprise Tester security code for data entered at the CA Enterprise Tester Code Security Panel, and possibly for a user ID other than the user ID that entered the request, completed successfully. When the message is issued, "code" is replaced by the encoded password value and may be used by the user ID for whom it was encoded with the TYPE statement CODE keyword to enter the data encoded.

PTE00122 - USER ID MUST BE 2 THROUGH 8 NATIONAL CHARACTERS

A request to create a CA Enterprise Tester security code for data entered at the CA Enterprise Tester Code Security Panel, and possibly for a user ID other than the user ID that entered the request, could not be processed because the active or specified user ID was not 2 through 8 national characters (A-Z, 0-9, @, # or $).

PTE00123 - DATA TO ENCODE MUST BE AT LEAST 3 NATIONAL CHARACTERS

A request to create a CA Enterprise Tester security code for data entered at the CA Enterprise Tester Code Security Panel could not be processed because the data was not 3 through 8 contiguous national characters (A-Z, 0-9, @, # or $).

PTE00124 - CODE FOR OTHER UID AUTHORITY REVOKED AT SYSTEM LIMIT

A request to create a CA Enterprise Tester security code for data entered at the CA Enterprise Tester Code Security Panel or using the PTEIS encode subroutine, and for use by a user ID other than the user ID that requested the security code could not be processed because the authority of the user to encode data for use by another user ID was revoked after encoding data for other user ID(s) more times than allowed by CA Enterprise Tester system options.

PTE00125 - SECURITY LIMIT EXCEEDED BY UID a ENCODING PASSWORD FOR UID b

A request by user ID "a" to encode data for use by user ID "b" exceeded the PTEOM CODE,FORLIM=limit value specified in CA Enterprise Tester system options. The message is sent via WTO with routing and descriptor codes also specified by CA Enterprise Tester system options. Refer to the *CA Enterprise Tester System Guide* for a complete description of password encoding limit purpose and options.

PTE00126 - A TOKEN COULD NOT BE ASSIGNED BECAUSE PROFILE LOAD FAILED

Password data entered while capturing a session and while CA Enterprise Tester Token Security was active was not successfully replaced by a security token because the user's CA Enterprise Tester PROFILE data set could not be loaded. The cause of the error may be a damaged PROFILE data set, invalid PROFILE data or an error explained by other messages issued when the error occurred. If the cause of the error is a damaged PROFILE data set, the data set may be deleted, after which the CA Enterprise Tester Token Security Panel may be used to reenter any token data lost when the PROFILE data set is deleted.

PTE00127 - A TOKEN COULD NOT BE CREATED BECAUSE ALL TOKEN NAMES ARE USED

Password data entered while capturing a session and while CA Enterprise Tester Token Security was active was not successfully replaced by a security token because all possible token names were already used to represent other values represented by password tokens. Because CA Enterprise Tester exhausts all possible combinations of alphabetic characters (A-Z) that fit in a particular non-display field before it runs out of token names, it is highly improbable that this problem can occur for a valid reason. If the problem occurs, contact CA Enterprise Tester technical support for assistance.

PTE00128 - PROFILE DATASET, data.set.name, CONTAINS INVALID DATA

CA Enterprise Tester PROFILE dataset, "data.set.name", contained data that was not valid for the user using the data set. A PROFILE data set is created automatically by CA Enterprise Tester for each CA Enterprise Tester user when first needed using options specified by the PTEOM PROFILE statement in the CA Enterprise Tester system options table, and is managed and updated by CA Enterprise Tester as required to maintain information private to the user for whom a PROFILE data set was created. CA Enterprise Tester PROFILE datasets may be moved to other volumes, and may be copied for backup purposes or restored. However, in no other case is it valid to manually allocate a PROFILE data set or to copy data to a PROFILE data set from any source other than a previous backup of a PROFILE data set created by CA Enterprise Tester for a particular user. If a PROFILE data set is damaged, a new PROFILE data set may be created by deleting the damaged data set, after which the user who owns the data set may use the CA Enterprise Tester Token Security Panel to recreate any lost PROFILE token data.

PTE00129 - NON-DISPLAY INPUT DATA WAS CAPTURED AS SECURITY TOKEN|CODE n

While using a CA Enterprise Tester session, data entered into a non-display field was captured as a CA Enterprise Tester password token named "n" or as a password code whose value is "n". This message is displayed to the user that entered the data only when PTEOM SECURE,NOTIFY=YES is specified in CA Enterprise Tester system options. Message PTE00130 is displayed following PTE00129.

PTE00130 - USER ID uid MAY USE THE TOKEN|CODE TO REKEY THE DATA ENTERED

This message is always displayed following message, PTE00129, and informs a user who entered non-display data that was captured as a CA Enterprise Tester password token or code, that the token or code may be used by user ID "uid" to reenter the actual data captured as a token or code to gain access to any system or data that is permitted by entering the captured non-display data.

PTE00901 - LOGON FAILED SLU=termid PLU=applid

PTECS was unable to successfully establish a session between a virtual terminal (SLU=termid) and an online system (PLU=applid). The caller of PTECS specified LOGONFAIL processing to request that in the event of a session connect error, that the console operator be given an opportunity to perform problem determination and correction, and the option to retry or reject the LOGON request. The message is preceded by another message that provides more specific information related to the cause of the problem. The message is issued as a non-deletable WTO. The message is followed by message PTE00902. The message is deleted from the console when PTE00902 is replied to.

PTE00902 - REPLY "RETRY", "DOWN" OR "REJECT"

PTE00902 follows PTE00901. The message is issued via WTOR. If the cause of the session initiation error can be determined and corrected, do so, and reply "RETRY". If the cause of the problem is that the online system is not active in the system, reply "DOWN". If the cause of the error can not be determined or corrected, reply "REJECT" to terminate the LOGON request. Possible causes of the error are: (1) VTAM is not active; (2) The PLU is not active; (3) the SLU is not active; (4) the SLU is "out of service" in the online system (PLU); (5) a cross domain resource manager or link connecting a PLU and SLU that reside in different network domains is inactive.

PTE00903 - LOGON RETRY REJECTED BY OPERATOR

PTECS was unsuccessful in initiating a CA Enterprise Tester session with a named online system. The CA Enterprise Tester script or Access Method application specified LOGONFAIL processing in the event that session initiation is unsuccessful. Console operator assistance was solicited via messages PTE00901 and PTE00902. The console operator replied "REJECT" to the request for corrective action. The return code is set to 8, and the message is returned to the caller.

PTE00904 - PLU IS DOWN PER CONSOLE OPERATOR

PTECS was unsuccessful in initiating a CA Enterprise Tester session with a named online system. The CA Enterprise Tester script or Access Method application specified LOGONFAIL processing in the event that session initiation is unsuccessful. Console operator assistance was solicited via messages PTE00901 and PTE00902. The console operator replied "DOWN" to the request for corrective action. The return code is set to 8, and the message is returned to the caller.

PTE10001 - PROTERM SCRIPT PROCESSOR STARTING - mm/dd/yy

This message is information only and is issued when every CA Enterprise Tester script is executed. The message documents the date that the script was executed.

PTE10002 - SCRIPT LOAD COMPLETE

This message is information only. The message indicates that the script was successfully loaded and that no syntax errors were found while parsing the loaded script(s).

PTE10003 - SCRIPT LOAD FAILED

One or more errors were encountered while loading and parsing the CA Enterprise Tester script that was executed. Previous messages document each error. The script is considered unexecutable. Correct the errors and restart the script.

PTE10004 - EXEC PARM IN ERROR

The parameters passed to the script processor via the JCL PARM field or via the Play Panel PARM field contain syntax errors. Refer to "Batch JCL" in topic 5.1 for a description of acceptable parameters and parameter formats. The script is considered unexecutable. Execute the script after correcting invalid script parameters.

PTE10005 - WTOR REPLY ACCEPTED: reply

This message is information only. A script, executed under the control of the Interactive System previously executed a PSL WTOR statement. The terminal user responded by entering the Interactive System REPLY command. In the message, 'reply', documents the data entered using the REPLY command.

PTE10006 - EXECUTING:

Either the script parameter, FLOW, or the PSL statement, FLOW ON, was specified or executed. When flow is active, the script processor logs each executed script verb just before it is executed. The script verb about to be executed is listed immediately following this message.

PTE10007 - STATEMENT EXECUTION LIMIT EXCEEDED

A PSL LIMIT statement was previously executed. The LIMIT statement specified that the script was to be terminated if any script verb was executed more times than a specified number. This message indicates that the limit has been exceeded.

PTE10010 - n STATEMENTS EXECUTED

This message is information only. When script execution ends, the message is issued to document the number of script verbs executed.

PTE10011 - systemname LOGON COMPLETED FROM terminalname

This message is information only. A PSL LOGON statement was executed to start a session with an online system. The session has been successfully started. When the message is issued, 'systemname' is replaced by the name of the online system, and 'terminalname' is replaced by the name of the CA Enterprise Tester virtual terminal used for the session.

PTE10012 - LOGON STATEMENT INTERNAL LOGIC ERROR

While executing a PSL LOGON statement, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10013 - TERMINAL NAME GT 8 CHARACTERS

The name of the CA Enterprise Tester virtual terminal specified following the FROM keyword in a PSL LOGON statement was more than 8 characters long. Specify a valid terminal name that is from 1 to 8 characters long.

PTE10014 - MODEL NAME GT 8 CHARACTERS

The terminal model name specified following the MODEL keyword in a PSL LOGON statement was more than 8 characters long. Specify a valid terminal model that is from 1 to 8 characters long.

PTE10015 - LOGON DATA GT 255 CHARACTERS

The logon data specified following the DATA keyword in a PSL LOGON statement was more than 255 characters long. Specify no more than 255 characters of LOGON data.

PTE10016 - ONLINE SYSTEM NAME VARIABLE IS EMPTY

The variable system that was found immediately following a PSL LOGON verb did not contain the name of an online system. When a variable system name is used, set the variable to a name before executing a LOGON statement.

PTE10017 - ONLINE SYSTEM NAME GT 8 CHARACTERS

The variable system name found following a PSL LOGON verb contained a value that was longer than 8 characters. When a variable name is specified, set the variable to a valid system name value before executing the LOGON statement.

PTE10018 - LOGON EXIT MESSAGE: message

While executing a PSL LOGON statement, and before attempting to start a session, a CA Enterprise Tester logon exit routine was called to process the logon request. The exit routine rejected the logon request. In the message, "message" is replaced by a message returned by the exit program. If the message is a CA Enterprise Tester message that begins with 'PTE', refer to the message for additional information, otherwise contact the systems programmer who installed the CA Enterprise Tester product.

PTE10019 - SIGNON EXIT MESSAGE: message

While executing a PSL LOGON statement, and just after starting a session, a CA Enterprise Tester signon exit routine was called to provide optional session start-up services. The exit routine returned a non zero return code. In the message, "message" is replaced by a message returned by the exit program. If the message is a CA Enterprise Tester message that begins with 'PTE', refer to the message for additional information, otherwise contact the systems programmer who installed the CA Enterprise Tester product.

PTE10021 - LOGON PROCESSING TERMINATED

An error was encountered while executing a PSL LOGON statement. A previous message describes the error. LOGON statement processing was not be completed as a result of the error.

The message is information only. Script execution proceeded at the next PSL statement. However, the presence of this message indicates that the online system being accessed sends delayed messages that should be anticipated by a script. A PSL INVITE statement, or a HANDLE CONTENTION statement may be used to receive or deal with delayed messages sent by online systems. "Session Flows" in topic 5.13describes applicable SNA protocol and techniques that allow a script to deal with online systems that initiate session brackets.

PTE10023 - A SESSION IS NOT ACTIVE

A PSL TYPE statement was executed, but a session was not active. A session must be started using a PSL LOGON statement prior to executing a TYPE statement to key data into a virtual terminal. If a session was previously started, it was also previously ended. A prior message would have been issued if a prior session has ended.

PTE10026 - CONVERSE EXIT MESSAGE: message

While executing a PSL LOGON statement, and just before sending keyed data to the online system, a CA Enterprise Tester converse exit routine was called to perform installation defined services. The exit routine returned a non zero return code. In the message, "message" is replaced by a message returned by the exit program. If the message is a CA Enterprise Tester message that begins with 'PTE', refer to the message for additional information, otherwise contact the systems programmer who installed the CA Enterprise Tester product.

PTE10028 - TYPE STATEMENT INTERNAL LOGIC ERROR

While executing a PSL TYPE statement, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10029 - AID VARIABLE DOES NOT CONTAIN A VALID AID KEY NAME

While executing a PSL TYPE statement that specified the AID keyword and a variable aid key, it was found that the variable field did not contain a valid AID key name. Valid AID key names are PF1 through PF24, PA1 through PA3, and ENTER. The AID variable field must be set to one of these values before executing the TYPE statement.

PTE10030 - TYPE OPERATION TERMINATED

An error was encountered while executing a PSL TYPE statement. A previous message describes the error. This message indicates that no additional TYPE statement processing was performed.

PTE10031 - CURSOR POSITIONED ON ATTRIBUTE AT ERASE EOF

While processing an ERASE keyword contained in a PSL TYPE statement, it was found that the cursor was currently positioned at a screen attribute location. The ERASE EOF operation could not be performed. Insure that the virtual terminal cursor is positioned only in an unprotected screen field when the ERASE operation is executed.

PTE10032 - CURSOR POSITIONED IN PROTECTED FIELD AT ERASE EOF

While processing an ERASE keyword contained in a PSL TYPE statement, it was found that the cursor was currently positioned in a protected screen field. The ERASE EOF operation could not be performed. Insure that the virtual terminal cursor is positioned only in an unprotected screen field when the ERASE operation is executed.

PTE10033 - VARIABLE "variable" TIME IS NOT HHMMSSTH

A PSL INVITE statement specified a variable time interval. In the message, "variable" is replaced by the name of the variable field. The variable did not contain a value of the form HHMMSSTH, expressing a time interval in hours, minutes, seconds, tenths and hundredths of seconds.

PTE10036 - INVITE NOT EXECUTED

An error was encountered while executing a PSL INVITE statement. A previous message describes the error. This message indicates that INVITE processing did not complete successfully.

PTE10037 - systemname LOGOFF COMPLETE AT:

This message is information only. A session with "systemname" has ended normally or abnormally. The message documents the script statement at which the session ended. If the session termination was abnormal, other messages provide information related to the session termination.

PTE10038 - SET STATEMENT INTERNAL LOGIC ERROR

While executing a PSL SET statement, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10039 - SOURCE DATA GT 32760 CHARACTERS

A SET statement was executed that specified more than 32760 characters of source data. A single CA Enterprise Tester variable field may hold no more then 32760 characters of data. The sum of the length of source data fields in a SET statement may not exceed this limit.

PTE10040 - SUBSTRING POSITION + LENGTH GT 32760

A PSL SET statement specified that a substring of a variable field was to be set to some source value. The sum of the target field substring position plus the length of the specified source data exceeded the maximum length of a single CA Enterprise Tester variable field.

PTE10041 - UP/DOWN TARGET VARIABLE NOT DECIMAL

The target field of a PSL SET statement that specified the UP or DOWN keyword was not a numeric variable or did not contain a single decimal value.

PTE10042 - SET UP/DOWN TARGET VARIABLE OVERFLOWED

A PSL SET statement was executed that attempted to set a target variable field UP or DOWN by a number that would have resulted in a decimal value that exceeded 30 significant integer digits.

PTE10043 - SET PROCESSING TERMINATED

An error was encountered while executing a PSL SET statement. A previous message describes the error. SET statement processing was not completed as a result of the error.

PTE10044 - COMPARE DATA INTERNAL LOGIC ERROR

While executing a PSL IF or DO statement, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10045 - ARG 1 INVALID - COMPARE NOT EXECUTED

A comparison specified in a PSL IF or DO statement could not be executed because the first argument in the relationship was invalid. A previous message describes the specific error.

PTE10046 - ARG 2 INVALID - COMPARE NOT EXECUTED

A comparison specified in a PSL IF or DO statement could not be executed because the first argument in the relationship was invalid. A previous message describes the specific error.

PTE10047 - MASK ON NOT POSSIBLE - MASK QUEUE IS EMPTY

A PSL MASK statement specified the keyword, ON, to activate previously specified MASK statement specifications. The operation could not be completed because no MASK statements have been previously executed or because queued MASK specifications were purged by a MASK RESET statement.

PTE10048 - GOTO NOT EXECUTED

A PSL GOTO statement could not be executed because a variable statement label or statement number was invalid. A previous message describes the specific error.

PTE10049 - PERFORM NOT EXECUTED

A PSL PERFORM statement could not be executed because a variable statement label or statement number was invalid. A previous message describes the specific error.

PTE10050 - VARIABLE SCRIPT NAME IS INVALID

A PSL LINK specified a variable script name. The value of the variable field was not a single 1 through 8 character script name.

PTE10051 - SCRIPT UNRESOLVED - INTERNAL LOGIC ERROR

While executing a PSL LINK statement, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10052 - LINK NOT EXECUTED

An error was encountered while executing a PSL LINK statement. A previous message describes the specific error.

PTE10053 - VARIABLE DOES NOT CONTAIN A VALID PROGRAM NAME

A PSL CALL specified a variable program name. The value of the variable field was not a single 1 through 8 character program name.

PTE10054 - INVALID PARM - INTERNAL LOGIC ERROR

While executing a PSL CALL statement, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10055 - CALL FAILED - ABEND Sxxx

While executing a PSL CALL statement, an z/OS LINK or LOAD macro instruction was issued to invoke the specified program. The instruction resulted in a system abend. In the message 'Sxxx' is replaced by the system abend code. Refer to IBM documentation of System Codes for an explanation of the abend. A common system code, S806, indicates that the named program could not be found.

PTE10056 - CALL PROCESSING TERMINATED

An error was encountered while executing a PSL CALL statement. A previous message describes the error. CALL statement processing was not completed as a result of the error.

PTE10058 - SCREEN NOT DUMPED - SESSION NOT ACTIVE

A PSL DUMP statement was executed that specified the SCREEN keyword. The screen could not be dumped because a session was not active. This message is information only. Script execution continues normally.

PTE10059 - SESSION SCREEN DUMP:

A PSL DUMP statement was executed that specified the SCREEN keyword, or an error has occurred and the PSL processor elected to dump the screen for possible use in determining the cause of the error. A dump of the CA Enterprise Tester virtual terminal screen follows this message.

PTE10060 - VARIABLE POOL DUMP:

A PSL DUMP statement was executed that specified no keywords or specified the POOL keyword, or an error has occurred and the PSL processor elected to dump the variable pool for possible use in determining the cause of the error. A dump of the currents of the CA Enterprise Tester variable pool follows this message.

PTE10061 - EXIT INVALID, NO PERFORM/LINK ACTIVE

A PSL EXIT statement was executed. The exit could not be processed because no return, to a statement that follows a previously executed PERFORM or LINK statement, is pending.

PTE10062 - LOG PROCESSING TERMINATED

An error was encountered while executing a PSL LOG statement. A previous message describes the error. LOG statement processing was not completed as a result of the error.

PTE10063 - "ddname" DD ALREADY OPEN AT OPEN

A PSL OPEN statement was executed that specified a ddname that was previously opened and has not yet been closed. When the message is issued, "ddname", is replaced by the ddname specified.

PTE10064 - OPEN INTERNAL LOGIC ERROR

While executing a PSL OPEN statement, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10065 - ABEND Sxxx OCCURRED WHILE OPENING "ddname"

A system abend occurred while executing a PSL OPEN statement. When the message is issued, 'Sxxx' is replaced by the system abend code and "ddname" is replaced by the ddname being opened. Refer to IBM documentation of System Codes for an explanation of the abend.

PTE10066 - SVC99 INFO ERROR, RC=xxxxxxxx

While executing a PSL OPEN statement, an attempt was made to obtain information about the data set being opened, using the z/OS dynamic information retrieval service. The information retrieval request completed unsuccessfully. When the message is issued, 'xxxxxxxx' is replaced by the SVC 99 error reason and information code. Refer to Appendix A of the help that is available when the HELP key is pressed while viewing the ISPF/PDF PRIMARY OPTION MENU or other IBM documentation of SVC 99 services for an explanation of the error.

PTE10067 - "ddname" IS A PDS, BUT MEMBER NOT SPECIFIED

While executing a PSL OPEN statement, it was found that the data set specified was a Partitioned Data Set, however, a PDS member name was not specified. CA Enterprise Tester data set access support may not be used to access an entire partitioned data set. When the message is issued, "ddname", is replaced by the ddname of the data set being opened.

PTE10068 - "ddname" OPEN FAILED - RC=cccc

An attempt to open the data set specified by a PSL OPEN statement was unsuccessful, or a VSAM open warning was issued and the statement did not specify the WARNING keyword. When the message is issued, 'cccc' is replaced by the VSAM open return code and ACB error code. Refer to IBM documentation of VSAN return and ACB error codes for an explanation of the error.

PTE10069 - "ddname" DATASET ORGANIZATION NOT SUPPORTED - reason

A PSL OPEN statement identified a data set whose organization is not supported by CA Enterprise Tester. When the message is issued, "ddname" is replaced by the ddname of the data set being opened and 'reason' is identifies the unsupported property. CA Enterprise Tester data set access supports processing of fixed and variable length sequential data sets, including partitioned data set members, and VSAM Key Sequenced data sets, with record lengths up to 32760 bytes. A VSAM KSDS must contain non-spanned records with unique keys.

PTE10070 - "ddname" DD IS NOT ALLOCATED

A PSL OPEN statement attempted to open a data set that could not be opened because the specified ddname was not allocated to the TSO or batch job address space. When the message is issued, "ddname", is replaced by the ddname specified.

PTE10071 - OPEN PROCESSING TERMINATED

An error was encountered while executing a PSL OPEN statement. A previous message describes the error. OPEN statement processing was not completed as a result of the error.

PTE10072 - "ddname" DD WAS NOT OPENED

A PSL READ statement attempted to access a data set that was not previously opened by executing a PSL OPEN statement. When the message is issued, "ddname" is replaced by the ddname specified. An OPEN statement must be issued prior to using other CA Enterprise Tester data set access statements.

PTE10073 - READ STATEMENT INTERNAL LOGIC ERROR

While executing a PSL READ statement, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10074 - "ddname" DD IS NOT OPEN

While executing a PSL READ statement it was found that the ddname specified was not open. An OPEN statement was previously executed, however, a previous end of file or other handled data set condition was encountered and the data set was not subsequently closed by executing a PSL CLOSE statement. After handling an unusual data set condition, close and reopen the data set prior to issuing other data set access statements. When the message is issued, "ddname", is replaced by the ddname specified.

PTE10075 - NON-KEYED READ EXECUTED AFTER ENDFILE, NOTFOUND OR ERROR

A PSL READ statement was executed that attempted to read a sequential data set after previously encountering an ENDFILE condition, or attempted to read a VSAM KSDS after previously encountering an ENDFILE, NOTFOUND or ERROR condition. The READ statement may not be used to access a sequential data set after ENDFILE. A READ statement used to access a VSAM KSDS, following an ENDFILE, NOTFOUND or ERROR condition, must specify a key.

PTE10076 - VSAM READ ERROR - RC/FDBK=cccc

While executing a PSL READ statement, a VSAM RPL error was encountered. When the message is issued, 'cccc', is replaced by the VSAM return code and error feedback code. Refer to IBM documentation of VSAM return and feedback error codes for an explanation of the error. VSAM conditions may be handled using the data set access ERROR condition keyword and by examining the PTERESP variable field.

PTE10077 - KEY SPECIFIED BUT DATASET IS NOT VSAM

A PSL READ statement was executed that specified a data set KEY, however, the data set is not a keyed data set. A keyed READ is supported only when accessing a VSAM KSDS.

PTE10078 - PREVIOUS SPECIFIED BUT DATASET IS NOT VSAM

A PSL READ statement was executed that specified the PREVIOUS keyword, however, the data set is not a VSAM KSDS. The PREVIOUS keyword may be used only while accessing a VSAM KSDS to read the data set in a backward.

PTE10079 - NOTFOUND SPECIFIED BUT DATASET IS NOT VSAM

A PSL READ statement was executed that specified the NOTFOUND condition. A NOTFOUND condition can occur only while executing a keyed read to a VSAM KSDS. The data set being accessed is not a VSAM KSDS.

PTE10080 - "ddname" DD IS OPEN FOR SEQUENTIAL OUTPUT ONLY

A PSL READ statement was executed to while accessing a sequential data set or a partitioned data set member. The data set may be read only opened using the INPUT keyword, however, the data set was currently opened for OUTPUT operations only. When the messages is issued, "ddname" is replaced by the ddname being accessed.

PTE10081 - SEQUENTIAL DSN MAY NOT BE READ FOLLOWING ENDFILE

A PSL READ statement was executed while accessing a sequential data set or partitioned data set member. An end of file condition occurred during the execution of a previous READ statement. After an end of file condition is encountered, a sequential data set or partitioned data set member must be closed and reopened before it may be read again.

PTE10082 - ABEND Sxxx OCCURRED WHILE READING "ddname"

A system abend occurred while executing a PSL READ statement. When the message is issued, 'Sxxx' is replaced by the system abend code and "ddname" is replaced by the ddname of the data set being read. Refer to IBM documentation of System Codes for an explanation of the abend.

PTE10083 - READ INTO VARIABLE WOULD EXCEED 32760 BYTES

A PSL READ statement was executed to read a record into a substring of a variable field. The maximum length of a CA Enterprise Tester variable field is 32760 bytes. The specified target substring position plus the length of the record read would exceed the limit.

PTE10084 - ENDFILE ENCOUNTERED BUT ENDFILE HANDLING NOT SPECIFIED

While executing a PSL READ statement, an end of file condition was encountered. The condition could not be handled because neither the READ statement nor the associated OPEN statement specified ENDFILE or ERROR condition handling.

PTE10085 - NOTFOUND SPECIFIED BUT A KEY HAS NOT BEEN SPECIFIED

A PSL READ statement specified the NOTFOUND keyword, but a KEY was not specified on this READ statement or on a previously executed READ statement. A NOTFOUND condition can be determined only when a full or generic key has been specified to support the decision.

PTE10086 - READ PROCESSING TERMINATED

An error was encountered while executing a PSL READ statement. A previous message describes the error. READ statement processing was not completed as a result of the error.

PTE10087 - "ddname" DD WAS NOT OPENED

A PSL WRITE statement attempted to access a data set that was not previously opened by executing a PSL OPEN statement. When the message is issued, "ddname" is replaced by the ddname specified. An OPEN statement must be issued prior to using other CA Enterprise Tester data set access statements.

PTE10088 - "ddname" DD IS OPEN FOR INPUT ONLY AT WRITE

A PSL WRITE statement was executed to write a record to a data set. The operation could not be performed because the data set was open for INPUT only when the WRITE statement was executed. When the message is issued, "ddname", is replaced by the ddname specified by the WRITE statement.

PTE10089 - WRITE FROM DATA LENGTH EXCEEDS 32760 BYTES

A PSL WRITE statement specified that a list of data items was to be concatenated to construct a record to be written. The operation could not be performed because the sum of the lengths of the data items specified exceeded 32760 bytes of data.

PTE10090 - WRITE FROM DATA LENGTH IS ZERO

A PSL WRITE statement was executed that specified one or more data items to be concatenated together to construct the record to be written. A record could not be constructed because the sum of the lengths of the data items specified was zero.

PTE10091 - WRITE STATEMENT INTERNAL LOGIC ERROR

While executing a PSL WRITE statement, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10092 - "ddname" DD IS NOT OPEN

While executing a PSL WRITE statement it was found that the ddname specified was not open. An OPEN statement was previously executed, however, a previously handled data set condition was encountered and the data set was not subsequently closed by executing a PSL CLOSE statement. After handling an unusual data set condition, close and reopen the data set prior to issuing other data set access statements. When the message is issued, "ddname", is replaced by the ddname specified.

PTE10093 - WRITE FROM DATA IS SHORTER THAN KEY POSITION + LENGTH

A PSL WRITE statement was executed to write a record to a VSAM KSDS. The length of the record specified was shorter than the sum of the data set key position plus key length. When a record is written to a VSAM KSDS, the record must be long enough to contain a full VSAM key.

PTE10094 - DUPKEY CONDITION ENCOUNTERED BUT DUPKEY NOT HANDLED

A PSL WRITE statement was executed to write a record to a VSAM KSDS. The record contained a key that already existed in the data set. The condition could not be handled because the WRITE statement did not specify the DUPKEY keyword, nor did the statement or the associated OPEN statement specify the ERROR keyword which may be used to handle duplicate keys.

PTE10095 - VSAM WRITE ERROR - RC/FDBK=cccc

While executing a PSL WRITE statement, a VSAM RPL error was encountered. When the message is issued, 'cccc', is replaced by the VSAM return code and error feedback code. Refer to IBM documentation of VSAM return and feedback error codes for an explanation of the error. VSAM conditions may be handled using the data set access ERROR condition keyword and by examining the PTERESP variable field.

PTE10096 - DUPKEY SPECIFIED BUT DATA SET IS NOT VSAM

A PSL WRITE statement was executed that specified handling for the DUPKEY condition. The vsam being accessed was not a VSAM KSDS and the duplicate key condition is not possible. The DUPKEY keyword may be specified only on WRITE statements that are used to write records to a VSAM KSDS.

PTE10097 - ABEND Sxxx OCCURRED WRITING TO "ddname"

A system abend occurred while executing a PSL WRITE statement. When the message is issued, 'Sxxx' is replaced by the system abend code and "ddname" is replaced by the ddname of the data set being written to. Refer to IBM documentation of System Codes for an explanation of the abend.

PTE10098 - WRITE PROCESSING TERMINATED

An error was encountered while executing a PSL WRITE statement. A previous message describes the error. WRITE statement processing was not completed as a result of the error.

PTE10099 - "ddname" DD WAS NOT OPENED

A PSL UPDATE statement attempted to access a data set that was not previously opened by executing a PSL OPEN statement. When the message is issued, "ddname" is replaced by the ddname specified. An OPEN statement must be issued prior to using other CA Enterprise Tester data set access statements.

PTE10100 - "ddname" DD IS OPEN FOR INPUT ONLY AT UPDATE

A PSL UPDATE statement was executed to update a record in a data set. The operation could not be performed because the data set was open for INPUT only when the UPDATE statement was executed. When the message is issued, "ddname", is replaced by the ddname specified by the UPDATE statement.

PTE10101 - UPDATE FROM DATA LENGTH EXCEEDS 32760 BYTES

A PSL UPDATE statement specified that a list of data items was to be concatenated to construct a record to be updated. The operation could not be performed because the sum of the lengths of the data items specified exceeded 32760 bytes of data.

PTE10102 - UPDATE FROM DATA LENGTH IS ZERO

A PSL UPDATE statement was executed that specified one or more data items to be concatenated together to construct the record to be updated. A record could not be constructed because the sum of the lengths of the data items specified was zero.

PTE10103 - UPDATE STATEMENT INTERNAL LOGIC ERROR

While executing a PSL UPDATE statement, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10104 - UPDATE IS SUPPORTED FOR VSAM ONLY

A PSL UPDATE statement was executed to update an existing record in a data set. The data set being accessed was not a VSAM KSDS. The UPDATE statement may be used to update records in a VSAM KSDS only.

PTE10105 - "ddname" DD IS NOT OPEN

While executing a PSL UPDATE statement it was found that the ddname specified was not open. An OPEN statement was previously executed, however, a previously handled data set condition was encountered and the data set was not subsequently closed by executing a PSL CLOSE statement. After handling an unusual data set condition, close and reopen the data set prior to issuing other data set access statements. When the message is issued, "ddname", is replaced by the ddname specified.

PTE10106 - UPDATE FROM DATA IS SHORTER THAN KEY POSITION + LENGTH

A PSL UPDATE statement was executed to update a record in a VSAM KSDS. The length of the record specified was shorter than the sum of the data set key position plus key length. When a record is updated in a VSAM KSDS, the updated record must be long enough to contain a full VSAM key.

PTE10107 - RECORD TO UPDATE NOT FOUND BUT NOTFOUND NOT HANDLED

A PSL UPDATE statement was executed to update a record in a VSAM KSDS. The data set did not contain a record with a key that matched the key of the record specified by the UPDATE statement. The condition could not be handled because the NOTFOUND keyword was not specified on the UPDATE statement, nor did the UPDATE statement or the associated OPEN statement specify the ERROR keyword, which may also be used to handle situations in which an update record did not previously exist.

PTE10108 - VSAM UPDATE ERROR - RC/FDBK=cccc

While executing a PSL UPDATE statement, a VSAM RPL error was encountered. When the message is issued, 'cccc', is replaced by the VSAM return code and error feedback code. Refer to IBM documentation of VSAM return and feedback error codes for an explanation of the error. VSAM conditions may be handled using the data set access ERROR condition keyword and by examining the PTERESP variable field.

PTE10109 - UPDATE PROCESSING TERMINATED

An error was encountered while executing a PSL UPDATE statement. A previous message describes the error. UPDATE statement processing was not completed as a result of the error.

PTE10110 - "ddname" DD WAS NOT OPENED

A PSL DELETE statement attempted to access a data set that was not previously opened by executing a PSL OPEN statement. When the message is issued, "ddname" is replaced by the ddname specified. An OPEN statement must be issued prior to using other CA Enterprise Tester data set access statements.

PTE10111 - "ddname" DD IS OPEN FOR INPUT ONLY AT DELETE

A PSL DELETE statement was executed to delete a record in a data set. The operation could not be performed because the data set was open for INPUT only when the DELETE statement was executed. When the message is issued, "ddname", is replaced by the ddname specified by the DELETE statement.

PTE10112 - DELETE STATEMENT INTERNAL LOGIC ERROR

While executing a PSL DELETE statement, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10113 - DELETE KEY VALUE LENGTH IS ZERO

A PSL DELETE statement was specified to delete one or more records from a VSAM KSDS. The statement specified a key, however the length of the key value was zero. A DELETE KEY must contain at least one character of data.

PTE10114 - DELETE IS SUPPORTED FOR VSAM ONLY

A PSL DELETE statement was specified to delete one or more records from a specified data set. The data set was not a VSAM KSDS. The DELETE statement may be used only to delete records from a VSAM KSDS.

PTE10115 - "ddname" DD IS NOT OPENED

While executing a PSL DELETE statement it was found that the ddname specified was not open. An OPEN statement was previously executed, however, a previously handled data set condition was encountered and the data set was not subsequently closed by executing a PSL CLOSE statement. After handling an unusual data set condition, close and reopen the data set prior to issuing other data set access statements. When the message is issued, "ddname", is replaced by the ddname specified.

PTE10116 - RECORD TO DELETE NOT FOUND BUT NOTFOUND NOT HANDLED

A PSL DELETE statement was executed to delete one or more records from a VSAM KSDS. No record existed in the data set that matched the full or generic key specified by the DELETE statement. The condition could not be handled because the DELETE statement did not specify the NOTFOUND keyword, nor did the DELETE statement or the associated OPEN statement specify the ERROR keyword, which may also be used to handle record not found conditions.

PTE10117 - VSAM DELETE ERROR - RC/FDBK=cccc

While executing a PSL DELETE statement, a VSAM RPL error was encountered. When the message is issued, 'cccc', is replaced by the VSAM return code and error feedback code. Refer to IBM documentation of VSAM return and feedback error codes for an explanation of the error. VSAM conditions may be handled using the data set access ERROR condition keyword and by examining the PTERESP variable field.

PTE10118 - DELETE PROCESSING TERMINATED

An error was encountered while executing a PSL DELETE statement. A previous message describes the error. DELETE statement processing was not completed as a result of the error.

PTE10119 - REOPEN FOR "ddname" DD FAILED, RC=cccc

A PSL statement was executed to perform an operation on a VSAM KSDS. The VSAM data set was empty when it was originally opened for OUTPUT, and therefore was opened for sequential output operations only. To obtain random access capabilities required by the current data set access statement, the data set was closed and an attempt was made to reopen the data set for random processing. The open failed. When the message is issued, "ddname", is replaced by the ddname being processed, and 'cccc' is replaced by the VSAM open return code and ACB error code. Refer to IBM documentation of VSAM return and ACB error codes for an explanation of the error.

PTE10120 - "ddname" DATASET EMPTY FOLLOWING LOAD

A PSL statement was executed to perform an operation on a VSAM KSDS. The VSAM data set was empty when it was originally opened for OUTPUT, and therefore was opened for sequential output operations only. To obtain random access capabilities required by the current data set access statement, the data set was closed and reopened to support direct access. After reopening the data set, it was found that the data set was empty. When the message is issued, "ddname", is replaced by the ddname being processed. Because CA Enterprise Tester attempts to reopen a data set only after records have been written, this error should never happen. Report the problem to CA Enterprise Tester technical support for assistance.

PTE10121 - "ddname" DD NOT OPEN AT CLOSE

A PSL CLOSE statement was executed to close a data set. The ddname specified had either never been opened by executing a PSL OPEN statement or CLOSE statement for the data set was previously executed. When the message is issued, "ddname", is replaced by the ddname being processed.

PTE10122 - CLOSE INTERNAL LOGIC ERROR

While executing a PSL CLOSE statement, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10123 - ABEND Sxxx OCCURRED WHILE CLOSING "ddname"

A system abend occurred while executing a PSL CLOSE statement. When the message is issued, 'Sxxx' is replaced by the system abend code and "ddname" is replaced by the ddname of the data set being closed. Refer to IBM documentation of System Codes for an explanation of the abend.

PTE10124 - CLOSE PROCESSING TERMINATED

An error was encountered while executing a PSL CLOSE statement. A previous message describes the error. CLOSE statement processing was not completed as a result of the error.

PTE10125 - VARIABLE FIELD DOES NOT CONTAIN A DDNAME

A PSL data set access statement was executed that specified a variable ddname. The variable field specified did not contain a ddname.

PTE10126 - VARIABLE DDNAME VALUE EXCEEDS 8 CHARACTERS

A PSL data set access statement was executed that specified a variable ddname. The variable field specified contained a value that was longer than 8 characters. A ddname must be a 1 through 8 character value that conforms to z/OS requirements.

PTE10127 - DDNAME VARIABLE CONTAINS MORE THAN ONE "WORD"

A PSL data set access statement was executed that specified a variable ddname. The variable field specified contained a more than one name value. A ddname variable must contain a single 1 through 8 character value that conforms to z/OS ddname requirements.

PTE10128 - VARIABLE VALUE "ddname" IS NOT A VALID DDNAME

A PSL data set access statement was executed that specified a variable ddname. The variable field specified contained a value that was not a valid z/OS ddname. When the message is issued, "ddname" is replaced by the variable value specified.

PTE10129 - OUTSTANDING WTO LIMIT EXCEEDED

To guard against "console lockout" resulting from a script getting into a runaway loop that contains a WTO statement, CA Enterprise Tester supports a WTO limit. The limit is specified via an option specified when CA Enterprise Tester is installed. When a limit is specified, no more than the specified number of WTO statements may be executed prior to executing a WTOR statement. The WTOR statement deletes previous WTO messages and resets the count. This message is issued when the WTO limit is exceeded.

PTE10130 - WTO PROCESSING TERMINATED

An error was encountered while executing a PSL WTO statement. A previous message describes the error. WTO statement processing was not completed as a result of the error.

PTE10131 - WTOR REPLY: reply

This message is information only. A PSL WTOR statement was executed to display a message on an z/OS console. The script was stopped until the console operator entered a reply to the message. This message indicates that the reply was received. When the message is issued, 'reply' is replaced by the reply entered by the console operator.

PTE10132 - WTOR MESSAGE CONTAINS NO DATA

A PSL WTOR statement was executed to send a message to a console operator or terminal user. The message specified one or more data items whose values were to be concatenated together to construct the message. The sum of the lengths of the values of the data items specified was zero. A WTOR statement must specify some message text.

PTE10133 - WTOR PROCESSING TERMINATED

An error was encountered while executing a PSL WTOR statement. A previous message describes the error. WTOR statement processing was not completed as a result of the error.

PTE10134 - HANDLE STATEMENT NOT PROCESSED

An error was encountered while executing a PSL HANDLE statement. A previous message describes the error. HANDLE statement processing was not completed as a result of the error.

PTE10135 - TRACE STATEMENT PROCESSING TERMINATED

An error was encountered while executing a PSL TRACE statement. A previous message describes the error. TRACE statement processing was not completed as a result of the error.

PTE10137 - RECORD PROCESSING TERMINATED

An error was encountered while executing a PSL RECORD statement. A previous message describes the error. RECORD statement processing was not completed as a result of the error.

PTE10138 - "ddname" DD WAS NOT OPENED

A PSL TRACK statement attempted to access a data set that was not previously opened by executing a PSL OPEN statement. When the message is issued, "ddname" is replaced by the ddname specified. An OPEN statement must be issued prior to using other CA Enterprise Tester data set access statements.

PTE10140 - "ddname" DD DSORG IS NOT SEQUENTIAL OR PARTITIONED

A PSL TRACK statement was executed to read a screen image from a CA Enterprise Tester trace data set. A CA Enterprise Tester trace data set must be a sequential data set or a member of a partitioned data set. The data set specified was another data set organization. When the message is issued, "ddname", is replaced by the ddname specified.

PTE10141 - "ddname" DD IS OPEN FOR OUTPUT ONLY

A PSL TRACK statement was executed to read a screen image from a CA Enterprise Tester trace data set. The data set can not be read because the specified ddname was opened for OUTPUT. When the message is issued, "ddname" is replaced by the ddname specified by the TRACK statement.

PTE10142 - "ddname" DD WAS ACCESSED AFTER ENDFILE

A PSL TRACK statement was executed to read a screen image from a CA Enterprise Tester trace data set. An end of file condition was previously handled by the script. After an end of file condition is encountered, a trace data set must be closed and reopened before it can be read again. When the message is issued, "ddname", is replaced by the ddname specified by the TRACK statement.

PTE10143 - "ddname" DD IS NOT OPEN

While executing a PSL TRACK statement it was found that the ddname specified was not open. An OPEN statement was previously executed, however, a previous end of file or other handled data set condition was encountered and the data set was not subsequently closed by executing a PSL CLOSE statement. After handling an unusual data set condition, close and reopen the data set prior to issuing other data set access statements. When the message is issued, "ddname", is replaced by the ddname specified.

PTE10144 - TRACK DATA SET MUST BE A "FORMAT" TRACE WITH LRECL 135

A PSL TRACK statement was executed to read a screen image from a CA Enterprise Tester trace data set. The TRACK statement may be used only to process a CA Enterprise Tester trace that was created using the FORMAT option and only when the data set LRECL is at least 135 characters. The data set specified does not contain screen images that can be processed by the TRACK statement.

PTE10145 - ABEND Sxxx OCCURRED WHILE TRACKING "ddname"

A system abend occurred while executing a PSL TRACK statement. When the message is issued, 'Sxxx' is replaced by the system abend code and "ddname" is replaced by the ddname of the trace data set being read. Refer to IBM documentation of System Codes for an explanation of the abend.

PTE10146 - ENDFILE ENCOUNTERED FOR "ddname" DD BUT ENDFILE NOT HANDLED

While executing a PSL TRACK statement, an end of file condition was encountered. The condition could not be handled because neither the TRACK statement nor the associated OPEN statement specified ENDFILE or ERROR condition handling. When the message is issued, "ddname", is replaced by the actual ddname being accessed.

PTE10147 - TRACK PROCESSING TERMINATED

An error was encountered while executing a PSL TRACK statement. A previous message describes the error. TRACK statement processing was not completed as a result of the error.

PTE10148 - VARIABLE "variable" TIME IS NOT HHMMSSTH

A PSL WAIT statement specified a variable time interval. In the message, "variable" is replaced by the name of the variable field. The variable did not contain a value of the form HHMMSSTH, expressing a time interval in hours, minutes, seconds, tenths and hundredths of seconds.

PTE10149 - WAIT INTERVAL INVALID

An error was encountered while processing the time interval specified on a PSL WAIT statement. A previous message describes the error. WAIT statement processing was not completed as a result of the error.

PTE10150 - SCRIPT PAUSE, AWAITING INTERACTIVE SYSTEM COMMAND

A PSL PAUSE statement was executed while executing a script under the control of the CA Enterprise Tester Interactive System. Script execution is stopped and the message is logged for examination via the Interactive System Monitor panel. The script may be resumed by the terminal user by entering an Interactive System WAIT, STEP, or RESUME command.

PTE10151 - THINK STATEMENT LOGIC ERROR

While processing a THINK statement an internal logic error was detected. This error indicates a CA Enterprise Tester product problem. Contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE10152 - THINK STATEMENT PROCESSING TERMINATED

An error was detected while attempting to calculate the time in which transactions must be completed in order to achieve the transaction rate specified by parameters of a PSL THINK statement. A previous message indicates the exact cause of the error. The likely cause is inconsistent or unrealistic THINK statement parameters.

PTE10153 - SCRIPT ENDING NORMALLY

This message is information only. The message is issued at the end of script execution when no unhandled errors were encountered.

PTE10154 - VARIABLE RC NOT 0 THROUGH 4095

A PSL END statement was executed to end script execution. The END statement specified a variable return code. The value of the variable was not a number from 0 through 4095.

PTE10155 - END STATEMENT IN ERROR - ENDING ABNORMALLY

An error was encountered while processing a PSL END statement. A previous message describes the error. Script execution is ended abnormally.

PTE10156 - VARIABLE ABEND CODE NOT 0 THROUGH 4095

A PSL ABEND statement was executed to abnormally end script execution. The ABEND statement specified a variable abend code. The value of the variable was not a number from 0 through 4095.

PTE10157 - ABEND STATEMENT ERROR

While executing a PSL ABEND statement, an error was encountered while processing a variable abend code. A previous message describes the error. Abnormal script termination continues, however the default abend code, 4095, is used.

PTE10158 - SCRIPT REQUESTED ABEND

A PSL ABEND statement was executed to abnormally terminate script execution. Script execution ends abnormally, using the abend code specified by the script or the default abend code, 4095.

PTE10159 - SESSION PASSED FROM system1 TO system2

While executing a PSL LOGON, TYPE or INVITE statement, the active session was passed from 'system1' to 'system2'. This message is information only. Sessions are passed to other systems as a normally part of the logon process performed by many online systems, including TSO and NETVIEW. Other online systems, including the IBM Information Network, pass sessions to other systems when specific services are requested during a session.

PTE10160 - SESSION PASSED WITH RECONNECT OFF - TERMINATING SESSION

While executing a PSL LOGON, TYPE or INVITE statement, an online system attempted to pass an active session to another system. A HANDLE RECONNECT OFF statement was previously executed. The statement specifies that passed sessions are not acceptable to the script. The active session is terminated.

PTE10161 - SCRIPT ILLOGIC HANDLED - CONTINUING

An error was encountered while executing a script statement. Previous messages describe the error. The script had previously executed a HANDLE ILLOGIC ON or HANDLE ILLOGIC AT statement. Script is execution is continued as specified by the HANDLE ILLOGIC statement.

PTE10162 - SCRIPT ILLOGIC NOT HANDLED, ABENDING

An error was encountered while executing a script statement. Previous messages describe the error. The script did not specify how to handle the error. Script is execution is abnormally terminated with an abend code, 4095.

PTE10163 - SESSION FAILURE NOT HANDLED, ABENDING

During script execution, a session was unexpectedly lost. A HANDLE SESSIONFAIL statement that requested abnormal script termination in the event of an abnormal session termination. Previous messages provide information related to the session termination. Script execution is terminated abnormally.

PTE10164 - VARIABLE USER ID VALUE NOT ENCODABLE

A PSL ENCODE statement was executed that specified a user ID to be used as a key when encoding data. The variable field specified contained a value that was not 2 through 8 national characters (alphabetic and numeric characters and the symbols @, #, and $). The value may not be used as a key by the CA Enterprise Tester encoding mechanism.

PTE10165 - ENCODE PROCESSING TERMINATED

An error was encountered while executing a PSL ENCODE statement. A previous message describes the error. ENCODE statement processing was not completed as a result of the error.

PTE10170 - "TO ALL" SOURCE DATA LENGTH IS ZERO

A PSL SET statement was executed that specified the ALL keyword to propagate a source value through a target variable field or variable field substring. The operation could not be performed because the length of the source pattern is zero.

PTE10171 - SET IMAGE INTERNAL LOGIC ERROR

While executing a PSL SET IMAGE statement, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10172 - IMAGE TARGET WOULD EXCEED 32760 BYTES

A PSL SET statement was executed that specified that a target variable field substring was to be set to the image of specified screen source data. The maximum length of a CA Enterprise Tester variable field is 32760 bytes of data. The operation could not be performed because the substring position plus the length of the data would exceed the maximum variable field size.

PTE10173 - "TO HEX" SOURCE IS NOT HEXADECIMAL CHARACTERS

A PSL SET statement was executed to convert a string of characters to internal hexadecimal characters. The specified source data contained characters other than the EBCDIC characters 0 through 9 and A through F.

PTE10174 - "FROM HEX" SOURCE IS NOT CHARACTER DATA

A PSL SET statement was executed to convert a data item to external hexadecimal characters. The FROM HEX option supports only character source values. The source value specified was a number or a CA Enterprise Tester numeric variable field.

PTE10175 - "FROM HEX" TARGET WOULD EXCEED 32760 CHARACTERS

A PSL SET statement was executed that specified that a target variable field or variable field substring was to be set to a value that consists of displayable characters that represent the hexadecimal value of the specified source data. The maximum length of a CA Enterprise Tester variable field is 32760 bytes of data. The operation could not be performed because the resulting variable value would exceed the maximum variable field size.

PTE10176 - "TO/FROM HEX" SOURCE DATA LENGTH IS ZERO

A PSL SET statement was executed to convert data from or to internal or external hexadecimal form. The length of the source data specified was zero. Some data must be specified when the SET statement TO or FROM HEX options are used.

PTE10177 - "TO BITS" SOURCE IS NOT ZEROS AND ONES

A PSL SET statement was executed to convert a string of character zeros and ones to internal binary bits. The source data specified contained data other than character zeros and ones.

PTE10178 - "FROM BITS" SOURCE DATA IS NOT CHARACTER

A PSL SET statement was executed to convert a data item to a string of character zeros and ones that represent the bits contained in the source data. The FROM BITS option supports only character source values. The source value specified was a number or a CA Enterprise Tester numeric variable field.

PTE10179 - "SET BITS" TARGET WOULD EXCEED 32760 CHARACTERS

A PSL SET statement was executed that specified that a target variable field or variable field substring was to be set to a string of zeros and ones that represent the bits contained in a specified field of source data, or that a variable field substring was to be set to a string of bits represented by a string of character zeros and ones contained in a source field. The maximum length of a CA Enterprise Tester variable field is 32760 bytes of data. The operation could not be performed because the resulting variable value would exceed the maximum variable field size.

PTE10180 - "SET BITS" SOURCE DATA LENGTH IS ZERO

A PSL SET statement was executed to convert data from character zeros and ones to binary bits or to convert binary bits to character zeros and ones. The length of the source data specified was zero. Some data must be specified when the SET statement TO or FROM BITS options are used.

PTE10181 - "SET CAPITAL" SOURCE DATA IS NOT CHARACTER

A PSL SET statement was executed to convert a data item to uppercase or lowercase alphabetic characters. The TO or FROM CAPITAL option supports only character source values. The source value specified was a number or a CA Enterprise Tester numeric variable field.

PTE10182 - "SET CAPITAL" TARGET WOULD EXCEED 32760 CHARACTERS

A PSL SET statement was executed that specified that a target variable field or variable field substring was to be set to an uppercase or lowercase representation of a specified source value. The maximum length of a CA Enterprise Tester variable field is 32760 bytes of data. The operation could not be performed because the resulting variable value would exceed the maximum variable field size.

PTE10183 - "SET CAPITAL" SOURCE DATA LENGTH IS ZERO

A PSL SET statement was executed to convert a data item to uppercase or lowercase alphabetic characters. The length of the source data specified was zero. Some data must be specified when the SET statement is used to convert the case of data.

PTE10184 - SET ZONED SOURCE DATA LENGTH IS ZERO

A PSL SET statement was executed to convert a decimal number to zoned format or to convert zoned number to decimal format. The length of the source data specified was zero. Some data must be specified when the SET statement is used to convert the form of a number.

PTE10185 - SET TO ZONED SOURCE DATA IS NOT DECIMAL

A PSL SET statement was executed to convert a decimal number to zoned format. The source data is not a valid decimal number. A decimal number may contain a plus or minus sign, an integer, a decimal point and a decimal fraction. An integer may not exceed 30 significant digits.

PTE10186 - ZONED VALUE OVERFLOWED SET TARGET FIELD

A PSL SET statement was executed that specified a target substring length and the TO ZONED option to convert a decimal number to zoned format. The length of the significant zoned number exceeds the specified target field substring length.

PTE10188 - SET FROM ZONED SOURCE DATA IS NOT ZONED

A PSL SET statement was executed that specified the FROM ZONED option to convert a zoned number to decimal format. The source value was not a valid zoned number. A zoned number consists of up to 30 zoned decimal numbers. The zone of the last digit may be a valid zoned decimal sign, X'A' through X'F'.

PTE10189 - SET PACKED SOURCE DATA LENGTH IS ZERO

A PSL SET statement was executed to convert a decimal number to packed or to convert a packed number to decimal. The length of the source data specified was zero. Some data must be specified when the SET statement is pack or unpack a number.

PTE10190 - SET TO PACKED SOURCE DATA IS NOT DECIMAL

A PSL SET statement was executed to convert a decimal number to packed format. The source data was not a valid decimal number. A decimal number may contain a plus or minus sign, an integer, a decimal point and a decimal fraction. An integer may not exceed 30 significant digits.

PTE10192 - TO PACKED TARGET EXCEEDS 16 BYTES

A PSL SET statement was executed that specified the TO PACKED option to convert a decimal number to packed format. The SET statement also specified a target substring length greater then 16. The SET statement supports the creation of only 1 through 16 byte packed numbers.

PTE10193 - TO PACKED SOURCE VALUE EXCEEDS TARGET LENGTH

A PSL SET statement was executed that specified a target substring length and the TO PACKED option to convert a decimal number to packed format. When packed, the significant digits of the decimal value result in a packed number whose length is greater than the specified target substring length.

PTE10195 - FROM PACKED SOURCE EXCEEDS 16 BYTES

A PSL SET statement was executed that specified the FROM PACKED option to convert a packed number to decimal format. The SET statement supports packed numbers that are 1 through 16 bytes in length. The operation could not be performed because a specified packed source value exceeds the 16 byte limitation.

PTE10196 - FROM PACKED SOURCE IS NOT PACKED DATA

A PSL SET statement was executed that specified the FROM PACKED option to convert a packed number to decimal format. The source data was not valid packed data.

PTE10197 - SET BINARY SOURCE DATA LENGTH IS ZERO

A PSL SET statement was executed to convert a decimal number to binary or to convert a binary number to decimal. The length of the source data specified was zero. Some data must be specified when the SET statement is used to convert the form of a number.

PTE10198 - SET TO BINARY SOURCE DATA IS NOT DECIMAL

A PSL SET statement was executed to convert a decimal number to binary format. The source data was not a valid decimal number. A decimal number may contain a plus or minus sign, an integer, a decimal point and a decimal fraction. An integer may not exceed 30 significant digits.

PTE10200 - SOURCE VALUE IN BINARY EXCEEDS 8 BYTES

A PSL SET statement was executed that specified the TO BINARY option to convert a number to internal binary format. SET statement support for binary data is limited to 1 through 8 byte signed or unsigned binary numbers. The target substring length specified exceeds the SET statement limitation.

PTE10201 - SOURCE VALUE IN BINARY EXCEEDS TARGET FIELD LENGTH

A PSL SET statement was executed that specified a target substring length and the TO BINARY option to convert a decimal number to binary format. The operation could not be performed because the length of the binary value exceeds the length of the target field.

PTE10203 - BINARY SOURCE VALUE EXCEEDS 8 BYTES

A PSL SET statement was executed that specified the FROM BINARY option to convert a binary number to decimal format. SET statement support for binary data is limited to 64 bit signed or unsigned binary numbers. The source value specified exceeds the SET statement limitation.

PTE10204 - VARIABLE "variable" DOES NOT CONTAIN A VALID SCRIPT LABEL

A PSL statement was executed that specified a variable script label. The variable field did not contain a valid script label name. In the message, "variable", is replaced by the name of the variable field whose value was invalid.

PTE10205 - VARIABLE LABEL "label" IS NOT A VALID LABEL VALUE

A PSL statement was executed that specified a variable script label. The variable field did not contain a valid script label. A valid script label begins with an alphabetic or national character (@, #, $), and contains only alphabetic, numeric and national characters and hyphens. When the message is issued, "label", is replaced by the invalid script label value.

PTE10206 - VARIABLE LABEL "label" NOT FOUND

A PSL statement was executed that specified a variable script label. The label was not found in the script currently being executed. When the message is issued, "label", is replaced by the variable label value that could not be located.

PTE10207 - VARIABLE "VARIABLE" STATEMENT NUMBER NOT 1 - 2,147,483,647

A PSL statement was executed that specified a variable statement number. The variable field value was not an integer from 1 through 2,147,483,647. When the message is issued, "variable", is replaced by the name of the variable statement number field specified.

PTE10210 - VARIABLE SUBSCRIPT "variable" VALUE NOT 0 - 2,147,483,647

A variable field subscript was specified that contained a value other than an integer from 0 through 2,147,483,647. When the message is issued, "variable", is replaced by the name of the variable field that contained the invalid subscript value.

PTE10211 - VARIABLE POSITION "variable" VALUE NOT 1 - 32760

A variable field position was specified that contained a value other than 1 through 32760. When the message is issued, "variable", is replaced by the name of the variable field that contained the invalid position value.

PTE10215 - VARIABLE LENGTH "variable" VALUE NOT 1 - 32760

3A variable field length was specified that contained a value other than 1 through 32760. When the message is issued, "variable", is replaced by the name of the variable field that contained the invalid length value.

PTE10219 - "variable" SUBSTRING POSITION + LENGTH EXCEEDS 32760

A variable field substring expression was invalid because the substring position plus the substring length exceeded the maximum length of a single variable field. When the message is issued, "variable" is replaced by the name of the variable field whose substring expression was invalid.

PTE10220 - "variable" IS NUMERIC, SUBSTRING DISALLOWED

A substring expression was invalidly used to refer to a substring of a numeric variable field. When the message is issued, "variable", is replaced by the name of the numeric variable field.

PTE10221 - "variable" VARIABLE OR SUBSTRING HAS NO VALUE

A variable field or a variable field substring was used to supply a required data value. The variable field or substring contained no value. When the message is issued, "variable", is replaced by the name of the empty variable that was used where a value is required.

PTE10225 - VARIABLE "variable" IS NOT CHARACTER

A variable field was used to supply a required character value. The variable specified was not a character variable field. When the message is issued, "variable", is replaced by the name of the variable field that was invalidly used where a character value is required.

PTE10227 - TYPE DATA CONTAINS AN ATTRIBUTE (HEX VALUE 01 THRU 20)

A PSL TYPE statement was executed that attempted to key in a data item that contained a value from hexadecimal x'01' through x'20'. Within a virtual terminal screen, those values are reserved for use as screen field attributes. A TYPE statement may not be used to modify the format of a virtual terminal screen. Only an online system may place attributes in the terminal screen.

PTE10228 - UNABLE TO TYPE "c" OVER A SCREEN ATTRIBUTE

A PSL TYPE statement was executed that attempted to key a character of information at a screen location that contained a screen field attribute. When the message is issued, "c", is replaced by the character being keyed at the time of the error. The TYPE statement may be used to key data into unprotected screen fields only.

PTE10229 - ATTRIBUTE NOT FOUND - INTERNAL LOGIC ERROR

While executing a PSL TYPE statement, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10230 - UNABLE TO TYPE "c" IN A PROTECTED SCREEN FIELD

A PSL TYPE statement was executed that attempted to key data into a protected screen field. When the message is issued, "c", is replaced by the character being keyed at the time of the error. The TYPE statement may be used to key data into unprotected screen fields only.

PTE10231 - VIEW DATA TYPE INTERNAL LOGIC ERROR

While resolving a PSL statement reference to a virtual terminal screen field, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10232 - SCREEN NOT AVAILABLE - SESSION NOT ACTIVE

A PSL statement referred to a virtual terminal screen field. References to the virtual terminal screen are valid only while a session is active. When the statement was executed, a session was not active.

PTE10233 - TRACKED SCREEN NOT READ USING TRACK

A PSL statement used the TRACKED adjective to refer to a screen image that was obtained from a CA Enterprise Tester trace data set. A reference to a tracked screen is valid only after successfully execution of a TRACK statement. The PSL statement reference was executed before successful execution of a TRACK statement.

PTE10234 - SCREEN ROW NOT 1 THROUGH SCREEN ROWS

A PSL statement contained a reference to a virtual terminal screen field or to a TRACKED screen field. The screen field row number specified was invalid. A screen field expression may refer only to rows 1 through the number of rows contained in the screen image referred to.

PTE10236 - SCREEN COLUMN NOT 1 THROUGH SCREEN COLUMNS

A PSL statement contained a reference to a virtual terminal screen field or to a TRACKED screen field. The screen field column number specified was invalid. A screen field expression may refer only to columns 1 through the number of columns contained in the screen image referred to.

PTE10237 - SCREEN FIELD LENGTH NOT VALID

A PSL statement contained a reference to a virtual terminal screen field or to a TRACKED screen field. The screen field length value specified was invalid. The length specified in a screen field expression must be a non-negative integer value.

PTE10238 - SCREEN FIELD POSITION + LENGTH EXCEEDS SCREEN SIZE

A PSL statement contained a reference to an active session screen or to a TRACKED screen image. The screen field expression specified a length that extended beyond the end of the screen image. A screen field length may be no longer than the remainder of the screen image measured from the screen row and column specified.

PTE10239 - LINE(S) AT ERROR: (origin)

This message is issued to identify one or more script lines that contained an error. Previous messages describe the error. In the message, "origin" is replaced by the source of the script line(s). When the script lines were read from a data set, origin identifies the line number of the line(s) in error relative to the data set(s) or PDS member from which the source line(s) were read and the name of the data set or PDS member, or the ddname, PTEIN, for the main script executed by a CA Enterprise Tester batch job. The line(s) in error are listed following this message. The line numbers contained on listed lines are relative to all script lines that have been loaded for execution, including the main script, linked scripts and script lines entered using the Interactive System EXECUTE command.

PTE10240 - STATEMENT AT ERROR:

This message is issued to identify a script statement that was being parsed when an error was detected, or that was being executed when an error was encountered. A previous message describes the error. In the message, "origin" is replaced by the source of the script statement. When the script statement was read from a data set, origin identifies the line number of the statement relative to the data set(s) or PDS member from which the statement was read and the name of the data set or PDS member, or the ddname, PTEIN, for the main script executed by a CA Enterprise Tester batch job. The statement is listed following this message. The line numbers contained on listed lines are relative to all script lines that have been loaded for execution, including the main script, linked scripts and script lines entered using the Interactive System EXECUTE command.

PTE10241 - ASSEMBLED VERB EXCEEDS 32K

The script statement being parsed required more than 32K of memory to store the "object code" required to represent all statement parameters. The possibility that a PSL statement could be coded that required anywhere near 32K of memory is so remote that this message represents a probable CA Enterprise Tester logic error. Report the error to CA Enterprise Tester technical support for assistance.

PTE10242 - EXEC PARM IS: parameters

This message is information only. It is issued when script options or parameters are specified via the JCL PARM of a CA Enterprise Tester batch job or via the PARM field of the Interactive System Play Panel, to document the options and parameters specified when a script was executed. When the message is issued, 'parameters', is replaced by the options and parameters specified.

PTE10243 - PARM LENGTH EXCEEDS 100 CHARACTERS

The parameter field passed to the PSL processor exceed 100 characters. This should be impossible since both the JCL PARM field and the Play Panel PARM field are limited to or no larger than 100 characters. Report the message to CA Enterprise Tester technical support for assistance.

PTE10244 - PARM SYNTAX ERROR ENCOUNTERED

The data specified via the JCL PARM field of a CA Enterprise Tester batch job or the Play Panel PARM field contained unsupported options or parameters. Refer to "Batch JCL" in topic 5.1 for a description of PARM field options and parameters.

PTE10245 - PARM KEYWORD NOT RECOGNIZED

The data specified via the JCL PARM field of a CA Enterprise Tester batch job or the Play Panel PARM field contained an unsupported option. An option is a word that is separated from other PARM field data by commas and that is not followed by or contain an equal sign. Refer to "Batch JCL" in topic 5.1 for a description of supported options and the syntax of user defined script parameters.

PTE10246 - PARM MAY NOT SET A "PTE" VARIABLE

The JCL PARM field of a CA Enterprise Tester batch job or the Play Panel PARM field specified a script parameter requesting that a variable field whose name begins with the characters 'PTE', be initialized to some value. Variables whose names begin with PTE may not be set using a script parameter.

PTE10247 - PARM VARIABLE NAME IS INVALID

The JCL PARM field of a CA Enterprise Tester batch job or the Play Panel PARM field specified a script parameter requesting that a variable field be set to some value. The variable field name specified is invalid. Valid variable field names may include only alphabetic and numeric characters, national characters (@, #, $), and hyphens and must begin with an alphabetic or national character.

PTE10255 - LOADING SCRIPT FROM origin

This message is information only. The message is issued just before loading a script for execution. When the message is issued, 'origin', is replaced by the data set or ddname from which the script is to be loaded.

PTE10256 - ABEND Sxxx OCCURRED WHILE LOADING SCRIPT

A system abend occurred while loading a CA Enterprise Tester script for execution. When the message is issued, 'Sxxx' is replaced by the system abend code. The data set being read was specified on the PTEIN DD statement of a CA Enterprise Tester batch job or via the Play Panel if the script was executed using the CA Enterprise Tester Interactive System. Refer to IBM documentation of System Codes for an explanation of the abend.

PTE10258 - PTEIN DD NOT ALLOCATED

The JCL of a CA Enterprise Tester batch job did not include a PTEIN DD statement. A PTEIN DD statement is required and identifies the data set containing the script to be executed.

PTE10259 - PTEIN DATA SET NOT PS OR PO

The PTEIN DD statement contained in the JCL of a CA Enterprise Tester batch job specified a data set that whose organization was not sequential or partitioned. A CA Enterprise Tester script must be supplied to CA Enterprise Tester in a sequential data set, a partitioned data set member or as 'DD *' data.

PTE10260 - PTEIN DATA SET IS PO, BUT MEMBER NOT SPECIFIED

The data set specified via the PTEIN DD statement contained in the JCL of a CA Enterprise Tester batch job was a partitioned data set, however, the DD statement did not specify a PDS member name. When the PTEIN data set is a PDS, a member name must be specified.

PTE10261 - PTEIN DATA SET IS PS, BUT MEMBER NAME SPECIFIED

The data set specified via the PTEIN DD statement contained in the JCL of a CA Enterprise Tester batch job was a sequential data set, however, a PDS member name was also specified. When PTEIN identifies a sequential data set, a member name may not be specified.

PTE10262 - LOADING LINKED SCRIPT script":

This message is information only. The message is issued while processing a LINK statement to document the name of a script that is being loaded to resolve the statement. When the message is issued, "script", is replaced by the name of the Script Library member being loaded.

PTE10263 - ABEND Sxxx OCCURRED WHILE LINKING "script"

A system abend occurred while processing a PSL LINK statement. When the message is issued, 'Sxxx' is replaced by the system abend code and "script" is replaced by the name of the script specified by the LINK statement. The script was being read from a CA Enterprise Tester Script Library specified via the PTELIB DD statement of a CA Enterprise Tester batch job or via the Play Panel if the script was executed using the CA Enterprise Tester Interactive System. Refer to IBM documentation of System Codes for an explanation of the abend.

PTE10264 - LINK "script" FAILED - LIBRARY NOT USABLE

A script specified by a PSL LINK statement could not be loaded because the CA Enterprise Tester Script Library is not usable. When the message is issued, "script", is replaced by the name of the script being loaded. A Script Library becomes unusable when it can not be opened or following a data set access abend that occurred while processing a LINK statement. A previous message describes the original data set access error.

PTE10265 - LINK "script" FAILED - SCRIPT NOT FOUND

A script specified by a PSL LINK statement could not be found in the library(s) specified via the PTELIB DD statement contained in the JCL of a CA Enterprise Tester batch job or in the libraries specified via the Interactive System Play Panel. When the message is issued, "script", is replaced by the name of the script.

PTE10266 - LINK "script" FAILED - BLDL I/O ERROR

An I/O error occurred when a BLDL macro statement was issued to read the Script Library directory entry of a script specified by a PSL LINK statement. When the message is issued, "script', is replaced by the name of the Script Library member specified. Script Libraries are specified via the PTELIB DD statement of a CA Enterprise Tester batch job or via the Interactive System Play Panel. LINK statement processing is terminated. The script can not be loaded.

PTE10267 - LINK "script" FAILED - FIND I/O ERROR

An I/O error occurred when a FIND macro statement was issued to locate the first record of a script specified by a PSL LINK statement. When the message is issued, "script', is replaced by the name of the Script Library member specified. Script Libraries are specified via the PTELIB DD statement of a CA Enterprise Tester batch job or via the Interactive System Play Panel. LINK statement processing is terminated. The script can not be loaded.

PTE10268 - PROCESSING EXECUTE COMMAND:

An Interactive System EXECUTE command was entered to execute one or more specified script statements. The script statements are listed following this message. The message is issued to document processing specified by the EXECUTE command.

PTE10269 - SCRIPT NOT EXECUTABLE

An Interactive System EXECUTE command was entered to execute one or more specified script statements. The command can not be executed because errors occurred while loading the originally played script, that prevented successful initiation of the script processing environment. The EXECUTE command was not processed.

PTE10270 - NO VALID STATEMENT FOUND

An Interactive System EXECUTE command was entered to execute one or more specified script statements. No valid PSL statement was found in the command data entered. EXECUTE command processing is terminated without action.

PTE10271 - COMMAND NOT EXECUTED

An Interactive System EXECUTE command was entered to execute one or more specified script statements. One or more errors were found while processing the entered command data. A previous message describes the error(s). EXECUTE command processing is terminated. No command statements were executed.

PTE10272 - "THEN" NOT FOUND FOLLOWING "IF"

A PSL IF statement was being parsed prior to script execution. The verb following the IF condition(s) was not THEN, GOTO, PERFORM, LINK, or CALL. The keyword, THEN must precede other script statements that are to be executed when the conditions are true. If no statements are to be executed when conditions are true, THEN NEXT ELSE, may be coded to identify the statements to be executed when conditions are false.

PTE10273 - NUMBER FOUND WHILE SCANNING FOR LABEL OR VERB

While scanning script source text for a script label or verb, a number was unexpectedly encountered. After logging the error message, the script parsing routine skips all subsequent text, until a script label or verb is found.

PTE10274 - LITERAL FOUND WHILE SCANNING FOR LABEL OR VERB

While scanning script source text for a script label or verb, a literal was unexpectedly encountered. After logging the error message, the script parsing routine skips all subsequent text, until a script label or verb is found.

PTE10275 - VARIABLE FOUND WHILE SCANNING FOR LABEL OR VERB

While scanning script source text for a script label or verb, a variable field expression was unexpectedly encountered. After logging the error message, the script parsing routine skips all subsequent text, until a script label or verb is found.

PTE10276 - SYMBOL FOUND WHILE SCANNING FOR LABEL OR VERB

While scanning script source text for a script label or verb, a special symbol was unexpectedly encountered. After logging the error message, the script parsing routine skips all subsequent text, until a script label or verb is found.

PTE10277 - UNKNOWN DATA FOUND WHILE SCANNING FOR LABEL OR VERB

While scanning script source text for a script label or verb, an unrecognized text item was unexpectedly encountered. After logging the error message, the script parsing routine skips all subsequent text, until a script label or verb is found.

PTE10278 - UNKNOWN WORD FOUND WHILE SCANNING FOR LABEL OR VERB

While scanning script source text for a script label or verb, a word was found that is not a PSL verb. After logging the error message, the script parsing routine skips all subsequent text, until a script label or verb is found.

PTE10279 - SKIPPING TO NEXT LINE LABEL OR VERB

An error was encountered while parsing script source text prior to script execution. A previous message describes the error. The script parsing routine skips all text following the error identified, until a valid script label or verb is found.

PTE10280 - LABEL IN "DO" ROUTINE OR "ENDDO" NOT FOUND

While parsing script source text prior to script execution, a script label or the end of script source text was encountered. It was found that an ENDDO statement was not found that delimits a previously processed DO statement. Either the ENDDO statement was not coded, or the DO routine contained a labeled script statement. A DO routine may contain only unlabeled script statements.

PTE10281 - LABEL IN "IF" ROUTINE OR "ENDIF" NOT FOUND

While parsing script source text prior to script execution, a script label or the end of script source text was encountered. It was found that an ENDIF statement was not found that delimits a previously processed IF statement. Either the ENDIF statement was not coded, or an IF routine contained a labeled script statement. An IF routine may contain only unlabeled script statements.

PTE10282 - AND/OR DOES NOT FOLLOW IF/WHILE/UNTIL

A PSL AND or OR verb was found while parsing script source prior to script execution. AND or OR may be used only to continue the list of conditions specified in by an IF, DO WHILE or DO UNTIL statement. The AND or OR encountered is not contained within a valid condition list.

PTE10283 - CALL PROGRAM NAME MISSING OR INVALID

While parsing a PSL CALL statement, an invalid program name value was found following the CALL verb. The value found was not a valid literal, actual or variable program name. If a literal name was specified, the literal did not contain a single 1 through 8 character program name.

PTE10284 - CALL PROGRAM NAME GT 8 CHARACTERS

While parsing a PSL CALL statement, an invalid program name value was found following the CALL verb. The value was a literal or actual program name, however the program name exceeded 8 characters.

PTE10285 - CALL USING KEYWORD IS REDUNDANT

While parsing a PSL CALL statement, the USING keyword was found in CALL statement parameters more than one time. When a parameter list is passed to a called program, specify the USING keyword once, followed by all parameters to be included in the parameter list.

PTE10286 - CALL PARM NOT LITERAL OR VARIABLE

While parsing a PSL CALL statement, a data item was found in the list of parameters following the USING keyword, that was not a literal or variable value. Only literal or variable values may be used as parameters to be passed to a called program.

PTE10287 - LITERAL CALL PARM LENGTH IS ZERO

While parsing a PSL CALL statement, a literal was found in the list of parameters following the USING keyword. The length of the literal was zero. When a literal is used as a call parameter, the literal must contain at least one character of information.

PTE10288 - NO CALL PARM FOUND FOLLOWING USING

While parsing a PSL CALL statement, the USING keyword was found. However the value found following the USING keyword was not a CA Enterprise Tester data item. When the optional USING keyword is specified, at least one parameter must be specified following the keyword.

PTE10289 - CALL VL KEYWORD IS REDUNDANT

While parsing a PSL CALL statement, the VL keyword was found more than one time. When the VL keyword is used to request that the last parameter in a USING list be marked, specify the keyword one time only.

PTE10290 - VL SPECIFIED BUT NO CALL USING LIST

While parsing a PSL CALL statement, the VL keyword was found. The VL keyword is used to request that the last parameter in a using list be identified to the called program. However, the USING keyword was not specified. Specify the optional VL keyword only when the USING keyword is specified to supply a parameter list containing one or more data items.

PTE10291 - REDUNDANT CLOSE CONDITION HANDLING SPECIFIED

While parsing a PSL CLOSE statement, the ERROR condition handling keyword was found. However, the ERROR keyword was previously found within the same CLOSE statement. Specify the ERROR keyword only one time per CLOSE statement.

PTE10292 - A DELETE KEYWORD IS SPECIFIED MORE THAN ONE TIME

While parsing a PSL DELETE statement, a DELETE statement keyword was found more than one time. In a single DELETE statement, specify the keywords, KEY, NOTFOUND, or ERROR, no more than one time.

PTE10293 - DELETE KEY IS NOT A LITERAL OR VARIABLE

While parsing a PSL DELETE statement, the KEY keyword was found. However, the data following the KEY keyword was not a literal or variable data item. Only a literal or variable data item may be used to specify the full or generic key value of records to be deleted.

PTE10294 - A NUMERIC VARIABLE MAY NOT BE USED AS A DELETE KEY

While parsing a PSL DELETE statement, the KEY keyword was found. However, the variable data item found following the KEY keyword is a CA Enterprise Tester defined numeric variable. When a variable field is used to specify the key value of records to be deleted, only a character variable field may be used.

PTE10295 - LITERAL DELETE KEY LENGTH IS ZERO

While parsing a PSL DELETE statement, the KEY keyword was found. However, the data item found following the KEY keyword is a zero length literal. The key value of records to be deleted must contain at least one character.

PTE10296 - DELETE DOES NOT SPECIFY KEY KEYWORD

While parsing a PSL DELETE statement, it was found that the KEY keyword was not specified. A DELETE statement is used to delete one or more records from a VSAM KSDS, that match a specified key value. The KEY keyword is a required DELETE statement keyword and must be coded to identify a key value data item.

PTE10297 - REDUNDANT DUMP OPTIONS SPECIFIED

While parsing a PSL DUMP statement, a DUMP statement keyword was found more than one time. Specify optional DUMP statement keywords, POOL or SCREEN, no more than one time per DUMP statement.

PTE10298 - "ELSE" NOT WITHIN AN IF STATEMENT

While parsing a script, prior to script execution, an ELSE statement was found. The ELSE statement is invalid because an IF statement is not being parsed or because a DO routine contained within an IF routine was not delimited by an ENDDO statement.

PTE10299 - "ELSE" ALREADY FOUND FOR "IF"

While parsing a script, prior to script execution, an ELSE statement was found. The ELSE statement is invalid because a previous ELSE statement was found that identified the routine to be executed when the current IF statement conditions are false.

PTE10300 - "ENDDO" NOT WITHIN A DO ROUTINE

While parsing a script, prior to script execution, an ENDDO statement was encountered. The ENDDO statement is invalid because a DO statement is not being processed. Use the ENDDO statement only to identify the end of a DO routine.

PTE10301 - "ENDDO" FOUND WITHIN IF ROUTINE

While parsing a script, prior to script execution, an ENDDO statement was encountered. The ENDDO is invalid because it is contained within an IF routine, and a corresponding DO statement is not also contained within the IF routine. A DO routine may be coded within an IF routine, but the DO and ENDDO must be wholly contained within the IF routine. An IF statement and true/false routines may be coded within a DO routine, but must be delimited by an ENDIF prior to delimiting the DO routine with an ENDDO statement.

PTE10302 - "ENDIF" NOT WITHIN AN IF ROUTINE

While parsing a script, prior to script execution, an ENDIF statement was encountered. The ENDIF statement is invalid because an IF statement that specified the THEN keyword is not being processed. Use the ENDIF statement only to identify the end of an IF statement that specified the THEN keyword.

PTE10303 - "ENDIF" FOUND WITHIN DO ROUTINE

While parsing a script, prior to script execution, an ENDIF statement was encountered. The ENDIF is invalid because it is contained within a DO routine, and a corresponding IF statement is not also contained within the DO routine. An IF statement may be coded within an DO routine, but the IF and ENDIF must be wholly contained within the DO routine. A DO statement may be coded within a IF routine, but must be delimited by an ENDDO prior to delimiting the IF routine with an ENDIF statement.

PTE10304 - REDUNDANT OR CONFLICTING ENCODE SPECIFICATIONS

While parsing a PSL ENCODE statement, redundant or conflicting ENCODE statement keywords were found. Either one of the keywords, ON, OFF and FOR, were coded more than one time, or ON and OFF were both coded, or OFF and FOR were both coded.

PTE10305 - ENCODE USER ID NOT LITERAL OR VARIABLE

While parsing a PSL ENCODE statement, the FOR keyword was found. However, the expression found following the FOR keyword was not a literal or variable user id to be used to encode data during script execution. Only a literal or variable field may be used to supply a user id to be used by the encoding mechanism.

PTE10306 - ENCODE LITERAL USER ID IS NOT ENCODABLE

While parsing a PSL ENCODE statement, the FOR keyword and a literal user id was found. However the literal user id value was not 2 through 8 national characters (alphabetic and numeric characters and the symbols @, #, and $). The value may not be used as a key by the CA Enterprise Tester encoding mechanism.

PTE10307 - NO ENCODE SPECIFICATIONS FOUND

While parsing a PSL ENCODE statement, no ENCODE keywords were found. When an ENCODE statement is coded, at least one ENCODE keyword, ON, OFF or FOR, must be coded.

PTE10308 - FLOW OPTION NOT ON OR OFF

While parsing a PSL FLOW statement, no FLOW statement keyword was found. When a FLOW statement is coded, specify one of the keywords, ON or OFF, to specify whether or not statements executed after the FLOW statement are to be logged prior to execution.

PTE10309 - "TO" NOT FOUND FOLLOWING "GO" VERB

While parsing a PSL GO statement, the keyword TO was not found following the verb, GO. 'GO TO' is an optional spelling of the verb, GOTO. When GO is used, TO is required.

PTE10310 - HANDLE CONDITION NAME MISSING OR INVALID

While parsing a PSL HANDLE statement, the expression found following the HANDLE verb was not recognized. The value that follows the HANDLE verb must be one of the condition keyword names, SESSIONFAIL, LOGONFAIL, CONTENTION, OUTBOUND, RECONNECT or ILLOGIC.

PTE10311 - LIMIT STATEMENT COUNT MISSING OR INVALID

While parsing a PSL LIMIT statement, an invalid value was found following the LIMIT verb. Limit must be an unsigned integer from 0 (zero) through 2,147,483,647.

PTE10313 - EXPECTED SCRIPT NAME NOT FOUND

While parsing a PSL LINK statement, the expression found following was not a valid script name. The value that follows the LINK verb must be a literal or a variable data item that specifies the Script Library member name to be performed.

PTE10314 - SCRIPT NAME NOT 1 THRU 8 CHARACTERS

While parsing a PSL LINK statement, an invalid actual or literal script name was found. The actual or literal value found following the LINK verb was not a single 1 through 8 character name of a Script Library member.

PTE10315 - ONLINE SYSTEM NAME MISSING OR INVALID

While parsing a PSL LOGON statement, an online system name was not found or an invalid online system name was found. The expression found following the LOGON verb was not a literal, actual or variable online system name or a literal online system name was specified and the length of the literal value was zero.

PTE10316 - ONLINE SYSTEM NAME GT 8 CHARACTERS

While parsing a PSL LOGON statement, an invalid online system name was found. The name found following the LOGON verb was an actual or literal value, however the name was more than 8 characters long.

PTE10317 - FROM TERMINAL NAME SPECIFIED TWICE

While parsing a PSL LOGON statement, the FROM keyword was found more than one time in the LOGON statement parameters. FROM is used to identify a data item as the name of the virtual terminal to be used for a session and may be specified only one time.

PTE10318 - FROM TERMINAL NAME MISSING OR INVALID

While parsing a PSL LOGON statement, the FROM keyword was found. However, the expression found following the FROM keyword was not an actual, literal or variable data item used to specify a virtual terminal name.

PTE10319 - FROM TERMINAL NAME GT 8 CHARACTERS

While parsing a PSL LOGON statement, the FROM keyword was found. However, the expression found following the FROM keyword was an actual or literal value that was longer than 8 characters. A valid virtual terminal name is 1 through 8 characters long.

PTE10320 - TERMINAL MODEL SPECIFIED TWICE

While parsing a PSL LOGON statement, the MODEL keyword was found more than one time in the LOGON statement parameters. MODEL is used to identify a data item as the model of a virtual terminal to be used for a session and may be specified only one time.

PTE10321 - TERMINAL MODEL MISSING OR INVALID

While parsing a PSL LOGON statement, the MODEL keyword was found. However, the expression found following the MODEL keyword was not an actual, literal or variable data item used to specify a virtual terminal model.

PTE10322 - TERMINAL MODEL NAME GT 8 CHARACTERS

While parsing a PSL LOGON statement, the MODEL keyword was found. However, the expression found following the MODEL keyword was an actual or literal value that was longer than 8 characters. A valid terminal model name is 1 through 8 characters long.

PTE10323 - LOGON DATA SPECIFIED TWICE

While parsing a PSL LOGON statement, the DATA keyword was found more than one time in the LOGON statement parameters. DATA is used to identify a data item as data to be passed to an online system when a session is started and may be specified only one time.

PTE10324 - LOGON DATA MISSING OR INVALID

While parsing a PSL LOGON statement, the DATA keyword was found. However, the expression found following the DATA keyword was not an literal or variable data item used to supply logon data to be passed to an online system when the LOGON statement is executed.

PTE10325 - MASK SPECIFICATIONS ARE CONFLICTING OR REDUNDANT

While parsing a PSL MASK statement, an invalid combination of mask specifications were found. Refer to "MASK" in topic 6.33, for a description of valid mask statement specifications.

PTE10326 - MASK <,,L> IS MUTUALLY EXCLUSIVE WITH PICTURE

While parsing a PSL MASK statement, both a data picture and a screen location that specified a screen field length, were found. When a picture is specified, do not specify screen field length. When a screen field length is specified, do not specify a data picture.

PTE10327 - MASK STATEMENT DATA PICTURE IS INVALID

While parsing a PSL MASK statement, an invalid data picture was found. Refer to "MASK" in topic 6.33 for a description of valid MASK statement data pictures.

PTE10328 - MASK LOCATION OR PICTURE EXTENDS BEYOND SCREEN ROW

While parsing a PSL MASK statement, a mask picture or a screen field expression was found that would place the masked screen field out of the bounds of a single screen row. Only screen fields that exist within a single screen row may be masked.

PTE10329 - MASK STATEMENT SCREEN LOCATION EXPRESSION IS INVALID

While parsing a PSL MASK statement, a screen field description was found that would place the field outside the bounds of any terminal screen. Masked screen fields must exist wholly within a single screen row.

PTE10330 - "NEXT" NOT WITHIN AN IF STATEMENT

While parsing a script prior to script execution, a PSL NEXT statement was found that is invalid where used. The NEXT statement may be used within an IF statement routine to go to the ENDIF statement. The NEXT statement is invalid because it is was not in an IF statement routine or because it was within a DO routine contained in an IF routine.

PTE10331 - AN OPEN KEYWORD WAS SPECIFIED MORE THAN ONE TIME

While parsing a PSL OPEN statement, an OPEN statement keyword was found more than one time. In a single OPEN statement, specify the keywords, INPUT, OUTPUT, ERROR, ENDFILE, or WARNING, no more than one time.

PTE10332 - CONFLICTING OPEN KEYWORDS SPECIFIED

While parsing a PSL OPEN statement, conflicting OPEN statement keywords were found. The OPEN statement specified both the INPUT and OUTPUT keywords. Specify only one. When a VSAM KSDS is opened for OUTPUT, both input and output operations may be performed.

PTE10333 - OPEN DOES NOT SPECIFY INPUT OR OUTPUT

While parsing a PSL OPEN statement, neither of the keywords INPUT nor OUTPUT were found. One of the keywords must be specified. When a VSAM KSDS is opened for OUTPUT, both input and output operations may be performed.

PTE10334 - A READ KEYWORD IS SPECIFIED MORE THAN ONE TIME

While parsing a PSL READ statement, a READ statement keyword was found more than one time. In a single READ statement, specify the keywords, INTO, KEY, PREVIOUS, NEXT, ENDFILE, NOTFOUND or ERROR, no more than one time.

PTE10335 - CONFLICTING READ DIRECTION SPECIFIED

While parsing a PSL READ statement, both of the READ statement keywords, NEXT and PREVIOUS, were found. A READ statement may specify only one read direction.

PTE10336 - READ INTO VARIABLE NOT SPECIFIED

While parsing a PSL READ statement, an invalid expression was found following the INTO keyword. The value specified following the INTO keyword must be the name of the variable field into which a record is to be read when the READ statement is executed. The expression found was not a valid variable field expression.

PTE10337 - READ INTO VARIABLE MAY NOT BEGIN WITH "PTE"

A PSL READ statement specified an INTO variable whose name begins with the characters, "PTE". "PTE" variables are defined by CA Enterprise Tester and may not be directly altered by a CA Enterprise Tester script. An INTO variable must be specified whose name does not begin with "PTE".

PTE10338 - READ KEY IS NOT A LITERAL OR VARIABLE

While parsing a PSL READ statement, an invalid key value was found. The key value specified following the READ statement KEY keyword must be a literal or a variable data item.

PTE10339 - A NUMERIC VARIABLE MAY NOT BE USED AS A READ KEY

While parsing a PSL READ statement, an invalid key value was found. The key value specified was a CA Enterprise Tester defined numeric variable field. Only a literal or a character variable field may be specified following the READ statement KEY keyword to supply the key value of a record to be read.

PTE10340 - READ DOES NOT SPECIFY AN INTO VARIABLE

While parsing a PSL READ statement, the INTO keyword was not found. The INTO keyword and a variable field expression must be specified in a READ statement to instruct CA Enterprise Tester where to place the record when the READ statement is executed.

PTE10341 - REDUNDANT OR CONFLICTING RECORD SPECIFICATIONS

While parsing a PSL RECORD statement, conflicting or redundant RECORD statement specifications were encountered. The ON and OFF keywords are mutually exclusive. ON or OFF may be specified only once per RECORD statement. A RECORD statement may not specify OFF and also specify data to be recorded.

PTE10342 - RECORD KEYWORD FOUND WITHIN RECORD DATA

While parsing a PSL RECORD statement, a RECORD statement keyword was found within a list of data items to be concatenated together to form a record to be recorded. When a list of data items is specified in a RECORD statement, and RECORD statement keywords are also specified, code the keywords either before or following the complete list of data items.

PTE10343 - NO RECORD SPECIFICATIONS FOUND

While parsing a PSL RECORD statement, no RECORD statement keyword or data item specifications were found. A RECORD statement must specify at least one keyword or data item.

PTE10346 - RC NOT 0 THROUGH 4095

While parsing a PSL ABEND or END statement, an numeric abend code or return code was found that was other than an integer from 0 (zero) through 4095.

PTE10347 - TARGET VARIABLE NOT FOUND FOLLOWING SET

While parsing a PSL SET statement, an invalid expression was found following the SET verb. The value specified following a SET verb must be a variable field expression that specifies the variable field or variable field substring to be altered by the SET statement. The value found was not a valid variable field expression.

PTE10348 - "PTE" VARIABLE MAY NOT BE ALTERED USING SET

While parsing a PSL SET statement, a CA Enterprise Tester defined variable field was found following the SET verb. The purpose of variable fields that begin with the characters "PTE" are defined by CA Enterprise Tester and may not be directly altered by a script.

PTE10349 - TO/FROM/UP/DOWN NOT FOUND FOLLOWING SET VARIABLE

While parsing a PSL SET statement, a value other than one of the SET statement keywords, TO, FROM, UP or DOWN, was found following the target variable field expression. One of these keywords must be specified to tell CA Enterprise Tester what operation is to be performed on the target variable field.

PTE10350 - INVALID DATA FOUND FOLLOWING SET TO/FROM

While parsing a PSL SET statement, an invalid specification was found following the TO or FROM keyword. In a SET statement, the specification that follows a TO or FROM keyword must be a CA Enterprise Tester data item, the symbol '*', or a SET statement adjective.

PTE10351 - SET STATEMENT INTERNAL LOGIC ERROR

While parsing a PSL SET statement prior to script execution, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10352 - INVALID DATA FOUND FOLLOWING SET ADJECTIVE

While parsing a PSL SET statement, a value other than a literal, number, screen field or variable field was found following a SET statement adjective. A valid CA Enterprise Tester data item must follow a SET statement adjective.

PTE10353 - DATA AFTER UP/DOWN NOT A VALID DATA ITEM

While parsing a SET statement that specified that a variable field was to be set UP or DOWN by some value, a valid data item was not found following the UP or DOWN keyword or following the BY keyword if the BY keyword was specified.

PTE10355 - "TO ALL" SOURCE DATA LENGTH IS ZERO

A PSL SET statement was found that specified the ALL keyword to propagate a source value through a target variable field or variable field substring. The operation can not be performed because the length of the source pattern is zero.

PTE10356 - IMAGE NOT FOLLOWED BY A SCREEN FIELD EXPRESSION

While parsing a PSL SET statement that specified the IMAGE adjective, a value other than a screen field expression or a TRACKED screen field expression was found following the IMAGE adjective. The value specified following the IMAGE adjective must be a valid CA Enterprise Tester screen field expression.

PTE10357 - "TO LENGTH" TARGET MAY NOT BE A SUBSTRING

A PSL SET statement was found that specified the LENGTH option to place the length of a data item into a CA Enterprise Tester numeric variable. The statement specified a target variable substring expression. A substring expression may not be used to refer to a CA Enterprise Tester numeric variable.

PTE10358 - "TO HEX" SOURCE IS NOT HEXADECIMAL CHARACTERS

While parsing a PSL SET statement that specified the TO HEX option, a literal value was found following the HEX adjective, however the literal contained characters other than 0 through 9 and A through F. The SET statement TO HEX option requires a source data value that contains only hexadecimal characters.

PTE10359 - "FROM HEX" SOURCE IS NOT CHARACTER DATA

While parsing a PSL SET statement that specified the FROM HEX option, a number was found following the HEX adjective. The SET statement FROM HEX option may be used only to convert a character data item to EBCDIC hexadecimal characters.

PTE10360 - "TO/FROM HEX" INTERNAL LOGIC ERROR

While parsing a PSL SET statement prior to script execution, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10361 - "TO/FROM HEX" SOURCE DATA LENGTH IS ZERO

A PSL SET statement was found that specifies the TO or FROM HEX option to convert data to or from internal or external hexadecimal form. The length of the source data specified was zero. Some data must be specified when the SET statement TO or FROM HEX options are used.

PTE10362 - "TO BITS" SOURCE IS NOT CHARACTER ZEROS AND ONES

While parsing a PSL SET statement that specified the TO BITS option, a value was found following the BITS adjective that was not a string of character zeros and ones. Only a literal or character variable field may be used to supply the required string of source zeros and ones.

PTE10363 - "FROM BITS" SOURCE DATA IS NOT CHARACTER

While parsing a PSL SET statement that specified the FROM BITS option, a number was found following the HEX adjective. The SET statement FROM BITS option may be used only to convert a character data item to a string of character zeros and ones.

PTE10364 - "SET BITS" INTERNAL LOGIC ERROR

While parsing a PSL SET statement prior to script execution, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10365 - "SET BITS" SOURCE DATA LENGTH IS ZERO

A PSL SET statement was found that specified the TO or FROM BITS option to convert data from character zeros and ones to binary bits or to convert binary bits to character zeros and ones. The length of the source data specified was zero. Some data must be specified when the SET statement TO or FROM BITS options are used.

PTE10366 - "SET CAPITAL" SOURCE IS NOT CHARACTER DATA

While parsing a PSL SET statement that specified the TO or FROM CAPITAL option, a number was found following the CAPITAL adjective. The SET statement CAPITAL option may be used only to convert the alphabetic case of a character data item.

PTE10367 - "SET CAPITAL" SOURCE DATA LENGTH IS ZERO

A PSL SET statement was found that specified the TO or FROM CAPITAL option to convert a data item to uppercase or lowercase alphabetic characters. The length of the source data specified was zero. Some data must be specified when the SET statement is used to convert the case of data.

PTE10368 - "SET CAPITAL" INTERNAL LOGIC ERROR

While parsing a PSL SET statement prior to script execution, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10369 - SET ZONED SOURCE DATA LENGTH IS ZERO

A PSL SET statement was found that specified the TO or FROM ZONED option to convert a zoned number to decimal format or to convert a decimal number to zoned format. The length of the source data specified was zero. Some data must be specified when the SET statement is used to convert the form of a number.

PTE10370 - SET ZONED INTERNAL LOGIC ERROR

While parsing a PSL SET statement prior to script execution, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10371 - SET ZONED SOURCE DATA EXCEEDS 30 DIGITS

While parsing a PSL SET statement that specified the TO or FROM ZONED option, a source number was found that exceeded 30 digits. SET statement support for zoned decimal numbers is limited to 30 digit numbers.

PTE10372 - SET TO ZONED SOURCE DATA IS NOT DECIMAL

While parsing a PSL SET statement that specified TO ZONED, a literal value was found following the ZONED adjective. The literal value was not a single decimal number. A decimal number may contain a leading sign, a 30 digit integer that may be punctuated with commas, a decimal point and a decimal fraction.

PTE10373 - SET FROM ZONED SOURCE DATA IS NOT ZONED

While parsing a PSL SET statement that specified FROM ZONED, a literal value or number was found following the ZONED adjective. The value did not consist of valid zoned data. Zoned data includes the character numbers 0 through 9 (X'F0' - X'F9'). The high order nibble of the last digit of a zoned number may be a valid packed sign, X'A' through X'F'.

PTE10374 - "SET PACKED" INTERNAL LOGIC ERROR

While parsing a PSL SET statement prior to script execution, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10377 - SET TO PACKED SOURCE DATA IS NOT DECIMAL

A PSL SET TO PACKED statement was found with a literal source field, however the source literal did not contain a valid decimal value. A valid decimal may contain a leading sign, an integer value including commas, a decimal point and a decimal fraction. The integer may not exceed 30 significant digits.

PTE10379 - SET FROM PACKED SOURCE DATA NOT 1-16 PACKED BYTES

A PSL SET statement was found that specified the FROM PACKED option to convert a packed decimal number to a CA Enterprise Tester decimal value. The source data item did not contain a valid packed data value.

PTE10381 - "SET BINARY" INTERNAL LOGIC ERROR

While parsing a PSL SET statement prior to script execution, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10383 - SET TO BINARY SOURCE IS NOT DECIMAL

A PSL SET TO binary statement was found with a literal source field, however the source literal did not contain a valid decimal value. A valid decimal may contain a leading sign, an integer value including commas, a decimal point and a decimal fraction. The integer may not exceed 30 significant digits.

PTE10387 - SET FROM BINARY SOURCE NOT 1-8 BYTES

A PSL SET statement that specified the FROM BINARY option, specified a source field that was not 1 through 8 bytes long. The SET statement supports only signed or unsigned binary numbers from 1 to 8 bytes in length.

PTE10389 - THEN IS NOT WITHIN AN IF STATEMENT

While parsing a script, prior to script execution, a THEN statement was found. The THEN statement did not immediately follow a list of IF statement conditions. USE the THEN keyword only to identify a routine that is to be executed when IF statement conditions are true, and code THEN immediately following the conditions.

PTE10391 - PTE VARIABLE MAY NOT BE USED AS TIME

While parsing a PSL THINK, INVITE or WAIT statement, an invalid time interval was found. The value found was a CA Enterprise Tester defined variable. No CA Enterprise Tester variable field has a value that is suitable for use as a time interval. Only an actual time of the form hhmmssth, expressing time in hours, minutes, seconds, and tenths and hundredths of a second, or a character variable field that contains an equivalent value, may be used to specify a THINK, INVITE, or WAIT time interval.

PTE10392 - TIME NOT 1-8 DIGITS (HHMMSSTH)

While parsing a PSL THINK, INVITE or WAIT statement, an invalid time interval was found. The value found was an actual time, however, the time was not a valid 1 through 8 digit value. Only an actual time of the form, hhmmssth, expressing time in hours, minutes, seconds, and tenths and hundredths of a second, or a character variable field that contains an equivalent value, may be used to specify a THINK, INVITE or WAIT interval.

PTE10393 - REDUNDANT OR CONFLICTING TRACE SPECIFICATIONS

While parsing a PSL TRACE statement, a redundant or conflicting specification was found. No TRACE statement keyword may be specified more than one time. The keywords, ON and OFF, are mutually exclusive. The keywords, IMAGE, FORMAT and UPPERCASE, are mutually exclusive. The keyword OFF, may not be specified when a trace format or caption is specified.

PTE10394 - TRACE KEYWORD FOUND WITHIN CAPTION

While parsing a PSL TRACE statement, a data item was found following a TRACE statement keyword. However, other caption data items were also found preceding the keyword. When bith keywords and caption data items are specified in a TRACE statement, specify keywords either before or following a single list of data items.

PTE10395 - NO TRACE SPECIFICATIONS FOUND

While parsing a PSL TRACE statement, no valid TRACE statement specifications were found. Every TRACE statement must specify at least one TRACE statement keyword or a trace caption.

PTE10396 - A TRACK KEYWORD IS SPECIFIED MORE THAN ONE TIME

While parsing a PSL TRACK statement, a TRACK statement keyword was found more than one time. In a single TRACK statement, specify the keywords, PHYSICAL, LOGICAL, ENDFILE or ERROR, no more than one time.

PTE10397 - CURSOR ORDER - INTERNAL LOGIC ERROR

While parsing a PSL TYPE statement prior to script execution, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10398 - SECURITY CODE NOT LITERAL OR VARIABLE

While parsing a PSL TYPE statement, an invalid CODE value was found. The value following the TYPE statement CODE keyword must be a literal or character variable field that contains a CA Enterprise Tester Security Code. Neither value was found.

PTE10399 - AID KEY NOT ACTUAL, LITERAL OR VARIABLE

While parsing a PSL TYPE statement, the TYPE statement AID keyword was found. However, the value found following the AID keyword was not the name of an aid key or a literal or character variable field. When the AID key is specified, the following value must be an aid key name, or a literal or variable aid key name. Valid aid key names are PF1 through PF24, PA1 through PA3 and ENTER.

PTE10400 - LITERAL AID KEY GT 128 CHARACTERS

While parsing a PSL TYPE statement, the TYPE statement AID keyword was found. However, the value found following the AID keyword was a literal with a length greater than 128 characters. When the AID key is specified, the following value must be an aid key name, or a literal or variable aid key name. Valid aid key names are PF1 through PF24, PA1 through PA3 and ENTER.

PTE10401 - AID KEY NAME IS NOT A VALID AID KEY

While parsing a PSL TYPE statement, the TYPE statement AID keyword was found. However, the literal or actual value found following the AID keyword was not the name of an aid key. Valid aid key names are PF1 through PF24, PA1 through PA3 and ENTER.

PTE10402 - CURSOR POSITIONING ORDER MAY NOT SPECIFY LENGTH

While parsing a PSL TYPE statement, a screen field expression was found that contained a length expression. In a TYPE statement, a screen field expression is used to position the cursor, and may contain only screen row and column specifications. If a screen field is to be typed elsewhere in a screen, it may be placed in a variable field, and then the variable field may be typed.

PTE10403 - AN UPDATE KEYWORD IS SPECIFIED MORE THAN ONE TIME

While parsing a PSL UPDATE statement, an UPDATE statement keyword was found more than one time. In a single UPDATE statement, specify the keywords, FROM, NOTFOUND or ERROR, no more than one time.

PTE10404 - NO DATA FOUND FOLLOWING FROM KEYWORD

While parsing a PSL UPDATE statement, the UPDATE statement FROM keyword was found. However, a valid data item was not found following the FROM keyword. One or more valid data items must be specified following the FROM keyword to supply the record that is to replace a record with a matching key in the VSAM KSDS to be processed.

PTE10405 - UPDATE DOES NOT SPECIFY FROM KEYWORD

A PSL UPDATE statement was found that did not specify the FROM keyword. FROM is a required UPDATE statement keyword and is used to identify one or more following data items that are to be concatenated together to construct the record that is to replace a record with a matching key in the VSAM KSDS to be processed.

PTE10406 - WHILE/UNTIL DOES NOT FOLLOW DO

While parsing a script, prior to script execution, a WHILE or UNTIL verb was found. However, the preceding word was not DO. WHILE or UNTIL may be coded only immediately following a DO verb to identify one or more conditions that control the number of times a DO routine is executed.

PTE10407 - A WRITE KEYWORD IS SPECIFIED MORE THAN ONE TIME

While parsing a PSL WRITE statement, a WRITE statement keyword was found more than one time. In a single WRITE statement, specify the keywords, FROM, DUPKEY or ERROR, no more than one time.

PTE10408 - NO DATA FOUND FOLLOWING FROM KEYWORD

While parsing a PSL WRITE statement, the WRITE statement FROM keyword was found. However, a valid data item was not found following the FROM keyword. One or more valid data items must be specified following the FROM keyword to supply the record to be written.

PTE10409 - WRITE DOES NOT SPECIFY FROM KEYWORD

A PSL WRITE statement was found that did not specify the FROM keyword. FROM is a required WRITE statement keyword and is used to identify one or more following data items that are to be concatenated together to construct the record to be written.

PTE10410 - COMPARE HAS MORE THAN 255 LEFT PARENTHESIS

While parsing a list of conditions specified in a PSL IF or DO statement, more than 255 left parenthesis were found. CA Enterprise Tester support for nesting conditions to order condition evaluation is limited to 255 nested condition levels.

PTE10411 - 1ST COMPARE ARGUMENT MISSING OR INVALID

While parsing a condition specified following a PSL IF, WHILE, UNTIL, AND or OR verb, a valid data item was not found following the verb. Excepting parenthesis, the first argument specified following IF, WHILE, UNTIL, AND or OR, must be the first of two data items to be compared.

PTE10412 - COMPARE CONDITION MISSING OR INVALID

While parsing a condition specified following a PSL IF, WHILE, UNTIL, AND or OR verb, a valid relational mnemonic or symbol(s) were not found following the first of two data items to be compared. Valid relational mnemonics are GT, GE, EQ, NE, LT and LE. Valid relational symbols are one or a logical pair of the symbols >, <, =, ª (not), and \ (not).

PTE10413 - 2ND COMPARE ARGUMENT MISSING OR INVALID

While parsing a condition specified following a PSL IF, WHILE, UNTIL, AND or OR verb, a valid data item was not found following the relational symbol or mnemonic. The argument following a relational symbol or mnemonic in a compare must be the second of two data items to be compared.

PTE10414 - COMPARE HAS MORE THAN 255 RIGHT PARENTHESIS

While parsing a list of conditions specified in a PSL IF or DO statement, more than 255 right parenthesis were found. CA Enterprise Tester support for nesting conditions to order condition evaluation is limited to 255 nested condition levels.

PTE10415 - TRACKED ADJECTIVE NOT FOLLOWED BY SCREEN FIELD EXPRESSION

While parsing a script prior to script execution, the TRACKED adjective was found. However, the value found following the TRACKED adjective was not a valid screen field expression. The TRACKED adjective may be used only to specify that the following data item is a field in a screen image read using the PSL TRACK statement.

PTE10416 - EXPECTED DD NAME MISSING OR INVALID

While parsing a script, prior to script execution, a value was found following a PSL data set access statement that was not a valid ddname expression. The first argument following every CA Enterprise Tester data set access verb must be a valid ddname expression. A ddname may be specified by coding an actual or literal ddname or by specifying a variable field name that will contain a ddname when the statement is executed. The ddname must conform to z/OS ddname rules and may not begin with the characters, "PTE". Ddnames that begin with "PTE" are defined by CA Enterprise Tester and are reserved for future product expansion.

PTE10417 - A NUMERIC VARIABLE MAY NOT BE USED AS A DD NAME

While parsing a script, prior to script execution, a CA Enterprise Tester defined numeric variable field was found following a PSL data set access verb. The first argument following every CA Enterprise Tester data set access verb must be a valid ddname expression. A ddname may be specified by coding an actual or literal ddname or by specifying a variable field name that will contain a ddname when the statement is executed. The ddname must conform to z/OS ddname rules. A CA Enterprise Tester numeric variable field may not be used to specify a ddname because a number is not a valid ddname.

PTE10418 - EXPECTED LABEL NOT FOUND

While parsing a PSL statement or statement keyword that requires a statement label or statement number to be specified, an invalid argument was found where the statement label or number was required. In a PSL GOTO, PERFORM or HANDLE condition AT statement, or following a data set access condition handling keyword, a statement label or number may be expressed as an actual label or statement number or as a numeric or character variable field that will contain a statement label or number when the statement is executed. The value found where a statement label or number is required was not one of these.

PTE10419 - STATEMENT NUMBER NOT 1 - 2,147,483,647

While parsing a PSL statement or statement keyword that requires a statement label or statement number to be specified, a number was found. However, the number was 0 or greater than 2,147,483,647. When a statement number is specified, specify a real statement number.

PTE10420 - LABEL CONTAINS NO ALPHA OR NATIONAL

While parsing a PSL statement or statement keyword that requires a statement label or statement number to be specified, an apparent label was found. However, the label does not contain an alphabetic or national character (@, #, $) and is therefore invalid.

PTE10421 - SCREEN FIELD ROW NOT FOUND

While parsing a CA Enterprise Tester script, prior to script execution, a screen field expression was found. However, the first argument found following the screen field symbol, '<', was not a row number or variable field.

PTE10422 - SCREEN FIELD ROW SUBSTRING DISALLOWED

While parsing a CA Enterprise Tester script, prior to script execution, a screen field expression was found. However, the variable screen row specified contains a substring expression. Only a number or numeric variable field may be used to specify screen row and a numeric variable field may not be referred to using substring notation.

PTE10424 - SCREEN ROW NUMBER NOT 1 THROUGH 43

While parsing a CA Enterprise Tester script, prior to script execution, a screen field expression was found. A number was used to specify screen row. The number was 0, was greater than 43, or was not an integer. A row number must refer to a line on the terminal screen.

PTE10425 - SCREEN FIELD COLUMN NOT FOUND

While parsing a CA Enterprise Tester script, prior to script execution, a screen field expression was found. However, the second argument found following the screen field symbol, '<', was not a column number or variable field.

PTE10426 - SCREEN FIELD COLUMN SUBSTRING DISALLOWED

While parsing a CA Enterprise Tester script, prior to script execution, a screen field expression was found. However, the variable screen column specified contains a substring expression. Only a number or numeric variable field may be used to specify screen column and a numeric variable field may not be referred to using substring notation.

PTE10428 - SCREEN COLUMN NUMBER NOT 1 THROUGH 132

While parsing a CA Enterprise Tester script, prior to script execution, a screen field expression was found. A number was used to specify screen column. The number was 0, was greater than 132, or was not an integer. A column number must be within a single line of a screen row.

PTE10429 - SCREEN FIELD EXPRESSION DELIMITER NOT FOUND

An invalid screen field expression was found while parsing a CA Enterprise Tester script, prior to script execution. An argument was found that was not a valid part of a screen field expression, prior to finding the screen field delimiter, '>'.

PTE10430 - SCREEN FIELD LENGTH NOT FOUND

An invalid screen field expression was found while parsing a CA Enterprise Tester script, prior to script execution. A comma was specified following the screen column specification indicating that a screen field length follows, however the value found following the comma was not a valid screen field length.

PTE10431 - SCREEN FIELD LENGTH SUBSTRING DISALLOWED

While parsing a CA Enterprise Tester script, prior to script execution, a screen field expression was found. However, the variable screen field length specified contains a substring expression. Only a number or numeric variable field may be used to specify screen field length and a numeric variable field may not be referred to using substring notation.

PTE10433 - SCREEN FIELD LENGTH VALUE INVALID

While parsing a script prior to script execution, a screen field expression was found that specified an invalid screen field length. A screen field length must be an integer no larger than the maximum size of a screen field.

PTE10434 - "variable" SUBSCRIPT/SUBSTRING DELIMITER MISSING

An invalid variable field expression was found while parsing a CA Enterprise Tester script, prior to script execution. An argument was found that was not a valid part of a variable field subscript or substring expression prior to finding the right parenthesis, ')', that is required to delimit the subscript or substring expression that was being parsed. When the message is issued, "variable", is replaced by the name of the variable field whose subscript/substring expression was invalid.

PTE10435 - "variable" SUBSCRIPT VALUE INVALID

While parsing a script prior to script execution, an invalid variable subscript value was found. The number exceeded the maximum subscript value of 2,147,483,647 or was not an integer. When the message is issued, "variable", is replaced by the name of the variable field whose subscript exceeded the CA Enterprise Tester limitation.

PTE10437 - "variable" SUBSTRING POSITION INVALID

While parsing a variable field expression, prior to script execution, an invalid variable field substring position was found. Position must be an integer from 1 through 32760. When the message is issued, "variable", is replaced by the name of the variable field whose substring position was invalid.

PTE10439 - "variable" SUBSTRING LENGTH INVALID

While parsing a variable field expression, prior to script execution, an invalid variable field substring length was found. Position must be an integer from 1 through 32760. When the message is issued, "variable", is replaced by the name of the variable field whose substring length was invalid.

PTE10441 - "variable" SUBSCRIPT/SUBSTRING CLOSING PAREN MISSING

An invalid variable field expression was found while parsing a CA Enterprise Tester script, prior to script execution. An argument was found that was not a valid part of a variable field subscript or substring expression prior to finding the right parenthesis, ')', that is required to delimit the subscript or substring expression that was being parsed. When the message is issued, "variable", is replaced by the name of the variable field whose subscript/substring expression was invalid.

PTE10442 - "variable" SUBSTRING POSITION + LENGTH EXCEEDS 32760

While parsing a variable field expression, prior to script execution, an invalid variable field substring expression was found. The position specified plus the length specified exceeded 32760, the maximum length of a single CA Enterprise Tester variable field. When the message is issued, "variable", is replaced by the name of the variable field whose substring specification were invalid.

PTE10443 - "variable" HAS MORE THAN 64 SUBSCRIPTS

While parsing a script, prior to script execution, a variable field expression was found that specified more than 64 subscripts. CA Enterprise Tester support for multiple dimension variable arrays is limited to 64 dimensions. When the message is issued, "variable", is replaced by the name of the variable field whose subscripts were excessive.

PTE10444 - PTE VARIABLES MAY NOT BE SUBSCRIPTED

While parsing a script, prior to script execution, a reference to a CA Enterprise Tester defined variable field was found. The variable field expression specified one or more subscripts. No CA Enterprise Tester defined variable field is a variable array. Subscripts may not be used to refer to CA Enterprise Tester defined variable fields.

PTE10445 - PTE NUMERIC VARIABLE SUBSTRING DISALLOWED

While parsing a script, prior to script execution, a reference to a CA Enterprise Tester defined variable field was found. The variable field expression specified a substring expression. The CA Enterprise Tester defined variable referred to is defined to be a numeric variable field. Substring notation may not be used when referring to any numeric variable field.

PTE10446 - /* NOT DELIMITED BY */

While parsing a CA Enterprise Tester script, prior to script execution, the beginning of a comment, identified by the characters, '/*', was found. The end of the script was encountered before finding the comment delimiter characters, '*/'.

PTE10447 - "c" IS AN UNSUPPORTED CHARACTER

While parsing a CA Enterprise Tester script, prior to script execution, an unsupported character was found. The character was not contained within a literal field. When the message is issued, "c", is replaced by the unsupported character.

PTE10448 - SCRIPT LABEL PREVIOUSLY DEFINED

While parsing a CA Enterprise Tester script, prior to script execution, a duplicate script label was encountered. Script labels, including script labels contained in scripts loaded to resolve a LINK statement, must be unique.

PTE10449 - NUMBER MAY NOT BEGIN IN COLUMN 1

While parsing a CA Enterprise Tester script, prior to script execution, a number was found in the first position of a script record. Only a script label or a comment may be coded in the first position of script records.

PTE10450 - LITERAL CLOSING QUOTE NOT FOUND

While parsing a CA Enterprise Tester script, prior to script execution, a single or double quote was found, indicating the start of a literal value. A second occurrence of opening quote was not found on the same script record. A literal, including opening and closing quotes must be contained within a single script record. If a larger value is needed, the SET statement may be used to concatenate several literals into a large variable field.

PTE10451 - LITERAL MAY NOT BEGIN IN COLUMN 1

While parsing a CA Enterprise Tester script, prior to script execution, a literal was found in the first position of a script record. Only a script label or a comment may be coded in the first position of script records.

PTE10452 - SYMBOL MAY NOT BEGIN IN COLUMN 1

While parsing a CA Enterprise Tester script, prior to script execution, a symbol, other than a comment symbol, was found in the first position of a script record. Only a script label or a comment may be coded in the first position of script records.

PTE10454 - NAME NOT FOUND FOLLOWING VARIABLE SYMBOL

While parsing a CA Enterprise Tester script, prior to script execution, a variable field symbol, '&', was found. The value found at the next record position was not a valid variable field name. The variable symbol must be the first character of every field name or reference to a variable field. A variable field name my include alphabetic, numeric and national (@, #, $) characters, and hyphens. The first character, following the variable symbol, must be an alphabetic or national character.

PTE10455 - 1ST VARIABLE NAME CHARACTER NOT ALPHA, @, # OR $

While parsing a CA Enterprise Tester script, prior to script execution, a variable field symbol, '&', was found. The value found at the next record position was not a valid variable field name. The variable symbol must be the first character of every field name or reference to a variable field. A variable field name my include alphabetic, numeric and national (@, #, $) characters, and hyphens. The first character, following the variable symbol, must be an alphabetic or national character.

PTE10456 - "PTE" VARIABLES ARE RESERVED - PTExxx IS NOT DEFINED

While parsing a CA Enterprise Tester script, a variable field name was found that begins with the characters, "PTE". Variables whose names begin with "PTE", are defined by CA Enterprise Tester or are reserved for future product expansion. When the message is issued, "PTExxx", is replaced by the undefined CA Enterprise Tester variable name found. The variable may not be defined by a CA Enterprise Tester script.

PTE10457 - VARIABLE MAY NOT BEGIN IN COLUMN 1

While parsing a CA Enterprise Tester script, prior to script execution, a variable field expression was found in the first position of a script record. Only a script label or a comment may be coded in the first position of script records.

PTE10458 - NUMBER IS INVALID OR TOO BIG

While parsing a CA Enterprise Tester script, a number was found that was invalid or that contained an integer with more than 30 significant digits. A number may contain a 1 through 30 digit integer, a decimal point, and a 1 through 30 digit fraction. If a decimal point is coded, a decimal digit must precede or follow the decimal point. A number may contain commas (European decimal point or American integer punctuation) only when contained in quotes or in variable data. When periods are used in European integer punctuation a number may not begin with a leading period.

PTE10459 - LABEL "label" NOT FOUND

While parsing a CA Enterprise Tester script, a PSL statement was found that referred to a script label that is not contained in the script being executed or in any script that has been included in scripts loaded by LINK statements.

PTE10460 - NEXT STATEMENT NOT RESOLVED

While resolving the structure of a CA Enterprise Tester script, prior to script execution, a PSL NEXT statement or an internally generated implicit NEXT statement was not resolved to identify the location of an ENDIF statement. Because a prior error should have prevented this error from being detected, the message may represent a CA Enterprise Tester internal logic error. If the reason for the error is not apparent, contact CA Enterprise Tester technical support for assistance.

PTE10461 - IF/WHILE/UNTIL PARENTHESIS NOT BALANCED

While resolving the structure of a set of conditions specified by an IF, DO WHILE or DO UNTIL statement, prior to script execution, it was found that the parenthesis specified to control the order in which a list of conditions is evaluated, do not balance.

PTE10462 - IF/WHILE/UNTIL ACTION NOT FOUND

While resolving the action(s) to be taken during execution depending on the results of comparisons specified by IF, DO WHILE and DO UNTIL statements, the location of a script routine was found to be unresolved. Because an error should have been detected that prevented the detection of this error, this error message represents a probable CA Enterprise Tester internal logic error. If the cause of the error is not apparent, report the problem to CA Enterprise Tester technical support for assistance.

PTE10464 - EXECUTE INSERTED BEFORE LINE 1

An Interactive System EXECUTE command was entered that specified that one or more PSL statements be inserted into the script loaded for execution. The command was entered at the Monitor panel and specified an insert line, 0. The statements have been inserted into the loaded script before the first line of the loaded script.

PTE10465 - EXECUTE INSERTED AFTER:

An Interactive System EXECUTE command was entered that specified that one or more PSL statements be inserted into the script loaded for execution. The command was entered at the Monitor panel and specified the line number at which the statements were to be inserted. The statements were inserted after the statement that is listed following this message.

PTE10466 - EXECUTE COMMAND INSERT LINE INVALID

An Interactive System EXECUTE command was entered that specified that one or more PSL statements be inserted into the script loaded for execution. The command was entered at the Monitor panel and specified the line number at which the statements were to be inserted. The statement number specified was not found in the loaded script and is invalid.

PTE10467 - NO MESSAGE DATA FOUND FOLLOWING WTO/WTOR VERB

A PSL WTO or WTOR statement was found that specified no message data items. WTO and WTOR statements must specify a message.

PTE10468 - WTO/WTOR ROUTCDE OR DESC IS NOT A NUMBER FROM 1 THRU 16

While parsing a PSL WTO or WTOR statement, a value other than a number from 1 through 16, was found following a a ROUTCDE or DESC keyword.

PTE10469 - WTO/WTOR ROUTCDE OR DESC VALUE IS LISTED MORE THAN ONCE

While parsing a PSL WTO or WTOR statement, a number was found to have been specified twice following a ROUTCDE or DESC keyword. Multiple route or descriptor code numbers from 1 through 16 may be specified, but a given number may be specified only one time.

PTE10470 - NO NUMBER FOUND FOLLOWING ROUTCDE KEYWORD

While parsing a PSL WTO or WTOR statement, the ROUTCDE keyword was found, but at least one route code was not found following the ROUTCDE keyword. When the ROUTCDE keyword is specified, one or more route code numbers from 1 through 16 must be specified.

PTE10471 - REDUNDANT, CONFLICTING OR INVALID WTO KEYWORD FOUND

While parsing a PSL WTO statement, a keyword was found that was specified more than one time, conflicting keywords were specified, or a keyword was found that is not supported by the WTO statement.

PTE10472 - NO NUMBER FOUND FOLLOWING DESC KEYWORD

While parsing a PSL WTO or WTOR statement, the DESC keyword was found, but at least one descriptor code was not found following the DESC keyword. When the DESC keyword is specified, one or more descriptor code numbers from 1 through 16 must be specified.

PTE10473 - REDUNDANT, CONFLICTING OR INVALID WTOR KEYWORD FOUND

While parsing a PSL WTOR statement, a keyword was found that was specified more than one time, conflicting keywords were specified, or a keyword was found that is not supported by the WTOR statement.

PTE10474 - SPECIFY TRACK PHYSICAL OR LOGICAL, NOT BOTH

While parsing a PSL TRACK statement, both of the TRACK statement keywords, PHYSICAL and LOGICAL were specified. The keywords are mutually exclusive.

PTE10475 - REDUNDANT FREE CONDITION HANDLING SPECIFIED

While parsing a PSL FREE statement, the ERROR keyword was found more than one time. When error handling is specified for an unallocation error, specify the ERROR keyword one time only.

PTE10476 - AN ALLOCATE KEYWORD IS SPECIFIED MORE THAN ONE TIME

While parsing a PSL ALLOCATE statement, an ALLOCATE statement keyword was found more than one time. In a single ALLOCATE statement, specify the keywords, DATASET, MEMBER, SHR, OLD, PASS or ERROR, no more than one time.

PTE10477 - LITERAL DATASET NAME IS INVALID

While parsing a PSL ALLOCATE statement, a literal data set name was found following the DATASET keyword. However the literal value does not contain a valid z/OS data set name.

PTE10478 - DATASET IS NOT A LITERAL OR CHARACTER VARIABLE

While parsing a PSL ALLOCATE statement, an invalid data set name specification was found. A data set name, specified following the ALLOCATE statement DATASET keyword, may be specified as a literal value or using a character variable field. The argument specified was not a literal or character variable field.

PTE10479 - LITERAL MEMBER NAME IS INVALID

While parsing a PSL ALLOCATE statement, a literal member name was found following the MEMBER keyword. However the literal value does not contain a valid partitioned data set member name.

PTE10480 - MEMBER IS NOT A LITERAL OR CHARACTER VARIABLE

While parsing a PSL ALLOCATE statement, an invalid member name specification was found. A member name, specified following the ALLOCATE statement MEMBER keyword, may be specified as a literal value or using a character variable field. The argument specified was not a literal or character variable field.

PTE10481 - ALLOCATE DATASET NOT SPECIFIED

A PSL ALLOCATE statement was found that did not specify a data set name. The ALLOCATE statement supports the allocation of cataloged data sets or partitioned data set members. The DATASET keyword and a data set name are required ALLOCATE statement parameters.

PTE10482 - ALLOCATE MUST SPECIFY ONE AND ONLY ONE DISP, SHR OR OLD

A PSL ALLOCATE statement was found that did not specify a data set disposition or specified conflicting data set disposition keywords. An ALLOCATE statement must specify one and only one of the disposition keywords, SHR or OLD.

PTE10483 - "ddname" DD PREVIOUSLY ALLOCATED

A PSL ALLOCATE statement was executed that specified a ddname that had been previously allocated using an ALLOCATE statement or that was previously opened using a PSL OPEN statement. The ddname must be closed and freed prior to allocating another data set to the ddname specified. When the message is issued, "ddname", is replaced by the ddname specified by the ALLOCATE statement.

PTE10484 - ALLOCATE INTERNAL LOGIC ERROR

While parsing a PSL ALLOCATE statement prior to script execution, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10485 - INVALID DATA SET NAME SPECIFIED

A PSL ALLOCATE statement was executed that specified a literal or variable data set name, following the ALLOCATE statement DATASET keyword, that did not contain a valid z/OS data set name.

PTE10486 - INVALID MEMBER NAME SPECIFIED

A PSL ALLOCATE statement was executed that specified a literal or variable member name, following the ALLOCATE statement MEMBER keyword, that did not contain a valid partitioned data set member name.

PTE10487 - ALLOCATION ERROR, RC=xxxxxxxx

While executing a PSL ALLOCATE statement, an attempt to allocate the specified data set or partitioned data set member name, using z/OS dynamic allocation services, was unsuccessful. When the message is issued, 'xxxxxxxx' is replaced by the SVC 99 error reason and information code. Refer to Appendix A of the help that is available when the HELP key is pressed while viewing the ISPF/PDF PRIMARY OPTION MENU or other IBM documentation of SVC 99 services for an explanation of the error.

PTE10488 - ALLOCATE PROCESSING TERMINATED

An error was encountered while executing a PSL ALLOCATE statement. A previous message describes the error. ALLOCATE statement processing was not completed as a result of the error.

PTE10489 - "ddname" DD NOT CLOSED BEFORE FREE

A PSL FREE statement was executed that requested unallocation of a specified ddname. The specified ddname was previously opened using a PSL OPEN statement and had not yet been closed. The data set must be closed by executing a PSL CLOSE statement prior to executing a FREE statement to unallocate the ddname. When the message is issued, "ddname", is replaced by the ddname specified by the FREE statement.

PTE10490 - FREE INTERNAL LOGIC ERROR

While parsing a PSL FREE statement prior to script execution, a CA Enterprise Tester internal logic error was detected. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10491 - DD FREE ERROR, RC=xxxxxxxx

While executing a PSL FREE statement, an attempt to unallocate the specified ddname, using z/OS dynamic unallocation services, was unsuccessful. When the message is issued, 'xxxxxxxx' is replaced by the SVC 99 error reason and information code. Refer to Appendix A of the help that is available when the HELP key is pressed while viewing the ISPF/PDF PRIMARY OPTION MENU or other IBM documentation of SVC 99 services for an explanation of the error.

PTE10492 - FREE PROCESSING TERMINATED

An error was encountered while executing a PSL FREE statement. A previous message describes the error. FREE statement processing was not completed as a result of the error.

PTE10493 - SCREEN FIELD LENGTH "*" PERMITTED ONLY IN COMPARE

While parsing a script, prior to script execution, a screen field expression was found that specified a screen field length, '*'. A screen field length of '*', may be specified only when a screen field is compared to a literal value to specify that the length of the screen field being compared is the same as the length of the literal value.

PTE10494 - BOTH ARGUMENTS MAY NOT SPECIFY "*" AS LENGTH

While parsing a script, prior to script execution, it was found that two fields being compared in an IF or DO statement specified "*" as the length of a variable field substring or as the length of a screen field. "*", when used as a length in a compare specifies that the length of the other argument is to be used in place of "*". When two arguments are being compared, "*" may be used as the length of only one of the data items.

PTE10495 - CALL REUSE KEYWORD IS REDUNDANT

While parsing a PSL CALL statement, the CALL statement REUSE keyword was found more than one time. When a CALL statement specifies REUSE, specify the keyword only one time.

PTE10496 - VARIABLE LENGTH "*" PERMITTED ONLY IN COMPARE

While parsing a script, prior to script execution, a variable field substring expression was found that specified a substring length, '*'. A substring length of '*', may be specified only when a variable field substring is compared to a literal value to specify that the length of the substring being compared is the same as the length of the literal value.

PTE10497 - SESSION CLOCKS FOR n TRANSACTIONS:

This message is issued when a session is terminated or when a PSL DUMP statement is executed that specifies the CLOCKS keyword. The message documents transaction, response time and think time statistics for the session. In the message, the number of transactions that have occurred during the session replaces "n". For the purpose of measuring response time and think time, a "transaction" is defined to be the set of session messages that begin with a terminal input and end with the last outbound message that precedes the next terminal input. "Response time" is the amount of time that has elapsed from the time of a terminal input to the receipt of the online system message that ends a transaction. "Think time" is the amount of time that has elapsed since the end of a previous transaction and the terminal input that starts the next transaction. The number of transactions, "n", is one less than the number of terminal inputs. A table of response time and think time statistics is logged following this message. The smallest, average and largest response and think times included in the table are expressed in seconds and are accurate to 1/100th of a second.

PTE10498 - NO SESSION CLOCKS ARE AVAILABLE

A DUMP statement was executed that specified the CLOCKS keyword. The requested session statistics could not be logged either because a session was not active or because less than two terminal inputs had occurred on the session. When less than two terminal inputs have been entered, it is not possible to determine response time or think time between transactions.

PTE10499 - TRACE CLOCKS FOR n TRANSACTIONS, DD=ddname

This message is issued when a trace data set, that has just been processed using the PSL TRACK statement, is closed using the CLOSE statement. The message documents transaction, response time and think time statistics for the session represented by the trace data set. In the message, "n" is replaced by one less than the number of inbound screen images that were read from the trace data set. For the purpose of measuring response time and think time, a "transaction" is defined to be the set of session messages that begin with a terminal input and end with the last outbound message that precedes the next terminal input. "Response time" is the amount of time that has elapsed from the time of a terminal input to the receipt of the online system message that ends a transaction. "Think time" is the amount of time that has elapsed since the end of a previous transaction and the terminal input that starts the next transaction. A table of response time and think time statistics is logged following this message. The smallest, average and largest response and think times included in the table are based on time stamps contained in the inbound and outbound screen images read from the trace data set and are accurate to 1/100th of a second.

PTE10501 - MASK ROW VARIABLE VALUE NOT 1-43

While executing a PSL MASK statement that specified a variable screen row value, it was found that the variable value was not 1 through 43. If a variable screen row is specified in a MASK statement, the variable must contain a valid screen row when the MASK statement is executed.

PTE10502 - LAST MASK ROW IS LESS THAN FIRST MASK ROW

While executing a PSL MASK statement that specified a range of screen rows, it was found that the first row specified followed the last row specified. When a range of rows is specified, the first row specified must precede the last row specified.

PTE10503 - MASK COLUMN VALUE NOT 1-132

While executing a PSL MASK statement that specified a variable screen column value, it was found that the variable value was not 1 through 132. If a variable screen column is specified in a MASK statement, the variable must contain a valid screen column when the MASK statement is executed.

PTE10504 - MASK FIELD LENGTH NOT 1-132

While executing a PSL MASK statement that specified a variable screen field length, it was found that the variable value was not 1 through 132. A masked screen field must be wholly contained in a single screen row, and therefore may not exceed 132 characters.

PTE10505 - MASK FIELD OVERLAPS END OF SCREEN ROW

While executing a PSL MASK statement that specified either a screen field length or a data picture, it was found that the field, beginning at the column specified, extended beyond screen column 132. A masked screen field must be wholly contained within a single screen row.

PTE10506 - MASK STATEMENT PROCESSING TERMINATED

A MASK statement, listed following this message, was found to be invalid and was not executed. A message describing the MASK statement error was listed preceding this message.

PTE10507 - TO ZONED TARGET WOULD EXCEED 32760

A PSL SET statement was executed to convert a decimal number to zoned format. The statement specified a target field substring position. The maximum length of a CA Enterprise Tester variable field is 32760 bytes. The specified target substring position plus the length of the zoned value data exceeded the limit.

PTE10508 - TO PACKED TARGET WOULD EXCEED 32760

A PSL SET statement was executed to convert a decimal number to packed format. The statement specified a target field substring position. The maximum length of a CA Enterprise Tester variable field is 32760 bytes. The specified target substring position plus the length of the packed decimal data would have exceeded the limit.

PTE10509 - TO BINARY TARGET WOULD EXCEED 32760

A PSL SET statement was executed to convert a decimal number to binary format. The statement specified a target field substring position. The maximum length of a CA Enterprise Tester variable field is 32760 bytes. The specified target substring position plus the length of the binary data would have exceeded the limit.

PTE10510 - PARM OVERLAYED BY CALLED SUBROUTINE

A PSL CALL statement was executed that specified USING and one or more parameters. When the called program returned to the calling script, it was found that data following a data area that was supplied to the called program was overlaid. When a called subroutine will return values in data areas specified in a CA Enterprise Tester CALL statement, the script must use a variable field for the parameter and must insure that the length of the variable field is large enough to contain the largest value that may be returned by the called program. This may be accomplished using a SET statement to initialize the variable to sufficiently large value.

PTE10512 - LOGON TIMED OUT

A PSL LOGON statement was executed, but was interrupted prior to successfully starting a session. A LOGON statement may be interrupted in two ways. A script may execute a TIMEOUT statement to specify the maximum amount of time that CA Enterprise Tester should wait for the completion of any session related event, and may execute a HANDLE TIMEOUT statement to activate time out processing. When time out processing is active, and the time out interval expires while waiting on a single session event, LOGON statement execution is terminated. A LOGON statement may also be timed out when a user presses the Attention key or PA1 key, while waiting on a script LOGON statement to complete, and if the user specifies that the logon request is to be timed out. When a LOGON statement is timed out, processing continues according to the active HANDLE TIMEOUT specification. An additional LOGON statement must be executed if a session is to be started.

PTE10513 - LOGON TIMED OUT DURING SIGNON EXIT

A PSL LOGON statement was executed, but was interrupted prior to successfully starting a session. The session was successfully started, however was timed out while executing a user-written sign on exit routine in the installed CA Enterprise Tester Communications Exit Program. LOGON statement processing may be interrupted in two ways. A script may execute a TIMEOUT statement to specify the maximum amount of time that CA Enterprise Tester should wait for the completion of any session related event, and may execute a HANDLE TIMEOUT statement to activate time out processing. When time out processing is active, and the time out interval expires while waiting on a single session event, LOGON statement execution is terminated. A LOGON statement may also be timed out when a user presses the Attention key or PA1 key, while waiting on a script LOGON statement to complete, and if the user specifies that the logon request is to be timed out. When a LOGON statement is timed out during sign on exit processing, the session is terminated and processing continues according to the active HANDLE TIMEOUT specification. An additional LOGON statement must be executed if a session is to be started.

PTE10514 - VARIABLE PARM VALUE IS NOT DATASTREAM OR SCREEN SIZE

A PSL LOGON statement specified a variable parameter, however the value of the variable field was not null and contained a value other than one 3270 data stream level keywords, EXTDS or BASICDS, or one of the default or alternate screen size keywords, 24X80, 32X80, 43X80 or 27X132. Correct the LOGON statement or variable field value.

PTE10515 - DATASTREAM SPECIFIED MORE THAN ONCE

A PSL LOGON statement specified conflicting or redundant 3270 data stream level keywords, EXTDS or BASICDS. The statement may have specified actual keywords or may have specified variable fields that contained the keywords. Correct the LOGON statement by coding only one 3270 data stream level specification.

PTE10516 - MORE THAN TWO SCREEN SIZES SPECIFIED

A PSL LOGON statement specified more than two of the screen size keywords, 24X80, 32X80, 43X80 or 27X132. Actual keywords or variable fields that contained the keywords may have been specified. The first keyword specified is taken to be the default screen size to be used for a session, and the second keyword is taken to be the alternate screen size. Correct the LOGON statement by coding no more than two screen size specifications.

PTE10517 - KEYBOARD IS LOCKED

A PSL TYPE statement was executed while the virtual terminal keyboard was locked. A previous TYPE or INVITE statement timed out while the virtual terminal was waiting on a passed session or while data traffic was reset. In those states, all keys are disabled, including the ATTN key. When timeout processing is used, a script must determine the state of a virtual terminal, prior to executing a TYPE statement. Variable fields, PTEKB, and PTESTATE contain values that may be used for this purpose. An INVITE statement must be used to process the next session event when the keyboard is locked.

PTE10518 - KEYBOARD IS INHIBITED

A PSL TYPE statement was executed while the virtual terminal keyboard was in an inhibited state. The variable field PTEKB, contains the value, INHIBITED, and indicates that the next normal flow message on the session must be from the online system to the virtual terminal. The condition may arise, only when HANDLE OUTBOUND, TIMEOUT or EVENTS statements are used to control the session flow.

While the virtual terminal keyboard is inhibited, all keys are disabled, with the exception of the Attention key. A TYPE statement may be executed while the keyboard is in an inhibited state, only if it specifies ATTN and no other data or keywords. An INVITE statement may be used to process the next session event. The variable field, PTESTATE, contains a state name that indicates the reason for the inhibited keyboard.

PTE10520 - VARIABLE "name" TIME IS NOT HHMMSSTH

A PSL TIMEOUT statement was executed, that specified a time interval using a variable field. When the message is issued, "name" is replaced by the name of the variable specified. The TIMEOUT statement could not be processed because the value of the variable field contained a value, however the value was not a valid time. To correct the TIMEOUT statement, insure that the variable field contains a time interval, expressed in hours, minutes, seconds, tenths and hundredths of a second, in the format HHMMSSTH.

PTE10521 - TIMEOUT TIME NOT SET

A invalid PSL TIMEOUT statement was encountered. A previous message documents the nature of the error. This message indicates that the TIMEOUT statement was not successfully executed, and identifies the invalid TIMEOUT statement that follows the message.

PTE10522 - PACKED DATA ERROR - UNKNOWN TOKEN

While executing a PSL TRACK statement, a screen image was being read from a trace data set that was written using the ISPF packed data format. The data could not be unpacked because a "TOKEN" (control character) was encountered that is not supported by the CA Enterprise Tester unpack routine. If the trace data set was written by CA Enterprise Tester or using the ISPF PACK and SAVE commands, and the reason for the invalid data is not known, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE10523 - PACKED DATA ERROR - EXCESSIVE BLANKS

While executing a PSL TRACK statement, a screen image was being read from a trace data set that was written using the ISPF packed data format. A control character was encountered that represented a string of repeated blanks. However, the length indicated that blanks were to be repeated past the end of the record being unpacked. If the trace data set was written by CA Enterprise Tester or using the ISPF PACK and SAVE commands, and the reason for the invalid data is not known, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE10524 - PACKED DATA ERROR - EXCESSIVE REPEAT

While executing a PSL TRACK statement, a screen image was being read from a trace data set that was written using the ISPF packed data format. A control character was encountered that represented a string of repeated characters. However, the length indicated that the character was to be repeated past the end of the record being unpacked. If the trace data set was written by CA Enterprise Tester or using the ISPF PACK and SAVE commands, and the reason for the invalid data is not known, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE10525 - PACKED DATA ERROR - EXCESSIVE STRING

While executing a PSL TRACK statement, a screen image was being read from a trace data set that was written using the ISPF packed data format. A control character was encountered that identified a string of uncompressed data. However, the length indicated that the string extends past the end of the record being unpacked. If the trace data set was written by CA Enterprise Tester or using the ISPF PACK and SAVE commands, and the reason for the invalid data is not known, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE10526 - PACKED DATA ERROR - END TOKEN INVALID

While executing a PSL TRACK statement, a screen image was being read from a trace data set that was written using the ISPF packed data format. A "TOKEN" (control character) was encountered that identified the end of the packed data. However, the token is invalid because the record being unpacked at the time the token was encountered, is incomplete. If the trace data set was written by CA Enterprise Tester or using the ISPF PACK and SAVE commands, and the reason for the invalid data is not known, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE10527 - SESSION TIMED OUT, HANDLE TIMEOUT IS OFF

A session timed out while executing a PSL LOGON, TYPE or INVITE statement. However, the current HANDLE TIMEOUT specification is OFF. The PSL statement was apparently timed out by a terminal user who pressed the Attention key while waiting on the completion of the statement, and then selected the option to time the session out. Because the script is not handling timeouts, the timeout is considered to be an error.

PTE10528 - SESSION FAILED, HANDLE SESSIONFAIL IS OFF

A session failure was detected by the event detection routine that was activated when a previous HANDLE EVENTS ON or AT statement was executed. A HANDLE SESSIONFAIL OFF statement was also previously executed, specifying that a session failure is to be treated as an error. This message informs the user that a session failure occurred.

PTE10529 - SESSION TIMED OUT, HANDLE TIMEOUT IS OFF

A session timed out while processing a session event, as instructed by a previously executed HANDLE EVENTS statement. However, the current HANDLE TIMEOUT specification is OFF. Event handling was apparently timed out be a terminal user who pressed the Attention key while waiting on the completion of an event, and then selected the option to time the session out. Because the script is not handling timeouts, the timeout is considered to be an error.

PTE10530 - ATTRIBUTES WOULD OVERFLOW SPECIFIED SUBSTRING

A PSL SET statement was executed that specified that the attributes of a virtual terminal screen location were to be placed in a substring of a target variable field. The operation could not be completed because the length of the string of attribute keywords would not fit in the specified target field substring.

PTE10531 - TARGET POSITION PLUS ATTRIBUTES EXCEEDS 32760 CHARACTERS

A PSL SET statement was executed that specified that the attributes of a virtual terminal screen location were to be placed at a particular position within a target variable field. The maximum length of a CA Enterprise Tester variable field is 32760 bytes. The SET statement could not be executed because the substring position plus the length of the string of attributes would exceed 32760 bytes.

PTE10532 - A TYPE CHARACTER IS INVALID (HEX 01 THRU 3F, NOT DUP OR FM)

A PSL TYPE statement was executed that attempted to key an invalid character into an extended data stream virtual terminal screen buffer. With the exception of the DUP character (X'1C') and Field Mark character (X'1E'), characters with hexadecimal values from X'01' through X'3F', have special meanings defined by 3270 architecture. A TYPE character was within the range of reserved values and could not be keyed into the virtual terminal screen.

PTE10533 - PTELOG OPEN FAILED - ABENDING

While preparing to execute a CA Enterprise Tester script, the PSL processor attempted to open a data set specified via the PTELOG DD statement. The attempted open failed. When executing as a batch job, the PSL processor requires a valid log data set to document events that occur during script execution. Previous messages written to the job log or z/OS SYSLOG, document the cause of the attach failure. Refer to the description of those messages and correct the cause.

PTE10536 - LOGON DATASTREAM CONFLICTING OR REDUNDANT

A PSL LOGON statement specified conflicting or redundant 3270 data stream level keywords, EXTDS or BASICDS. Correct the LOGON statement by coding only one 3270 data stream level specification.

PTE10537 - LOGON SCREEN SIZES CONFLICTING OR REDUNDANT

A PSL LOGON statement specified more than two of the screen size keywords, 24X80, 32X80, 43X80 or 27X132. The first keyword specified is taken to be the default screen size to be used for a session, and the second keyword is taken to be the alternate screen size. Correct the LOGON statement by coding no more than two screen size specifications.

PTE10538 - EXCESSIVE LOGON DATASTREAM/SIZE PARMS

A PSL LOGON statement was executed that specified a 3270 data stream level, BASICDS or EXTDS, more than once, or that specified virtual terminal screen sizes, 24X80, 32X80, 43X80 or 27X132, more than twice. Correct the LOGON statement by specifying only one 3270 data stream level and/or by specifying no more than two screen sizes.

PTE10539 - ATTRIBUTES MUST PRECEED SCREEN FIELD EXPRESSION

A PSL SET statement was executed that used the ATTRIBUTES adjective. The ATTRIBUTES adjective specifies that the target field is to be set to a list of keywords that describe the attributes of a screen location. The data following the ATTRIBUTES adjective was not a screen location in either the session screen or in a tracked screen. Correct the SET statement by specifying a valid screen expression, immediately following the ATTRIBUTES adjective.

PTE10540 - SPECIFY TRACK BASICDS OR EXTDS, NOT BOTH

A PSL TRACK statement specified more than one data stream level keyword. The keywords, BASICDS or EXTDS, may be used to specify that a screen image being read by the TRACK statement is to be converted to basic (a single screen buffer) or extended form (s screen buffer and an extended attribute buffer). Correct the TRACK statement by specifying no more than one of the keywords, BASICDS or EXTDS.

PTE10541 - PTEREC DD OPEN FAILED, SCRIPT NOT EXECUTED

A CA Enterprise Tester script was executed in a batch job using CA Enterprise Tester program, PTEBS. A PTEREC DD statement was present, indicating that a recording of any session created by the script was to be recorded. However, the target data set specified by the PTEREC DD statement was not a suitable target for a CA Enterprise Tester recording. A recording may be written to a data set, PDS member, or SYSOUT provided that the target is RECFM=FB and LRECL=80. A message that precedes this message states the problem with the target specified on the PTEREC DD statement. After this message is issued, the job step is terminated with a user 4095 abend. Specify a valid target data set or remove the PTEREC DD statement if a session recording is not desired.

PTE10542 - EXECUTE TEXT EXCEEDS 32760 BYTES

The expansion of the string of data items specified by a PSL EXECUTE statement exceeded 32760 bytes of data. When the values of the data items specified in an EXECUTE are substituted for the data item to construct the PSL statement to be executed, the resulting string of text may not exceed the 32760 byte limit.

PTE10543 - EXECUTE OPERAND IS NOT AN EXECUTABLE VERB

A PSL EXECUTE statement specified a CA Enterprise Tester verb that may not be executed using the EXECUTE verb. An EXECUTE statement may be used to execute a generated PSL statement that is not related to other statements. The following PSL verbs may not be executed: IF, AND, OR, THEN, NEXT, ELSE, ENDIF, DO, UNTIL, WHILE, ENDDO, LINK, EXECUTE.

PTE10544 - EXTRANEOUS DATA FOLLOWS EXECUTED STATEMENT

While processing a PSL EXECUTE statement, a valid PSL statement was constructed from the data items specified by the EXECUTE statement. However, concatenation of the data items specified by the EXECUTE statement yielded text, following a complete executable statement, that is not executable. A single EXECUTE statement may be used to execute only a single PSL verb and it's operands. Multiple execute statements are required to execute multiple generated PSL statements.

PTE10545 - TRACE PACK/NOPACK IS INVALID WHILE TRACE IS ACTIVE

While executing a PSL TRACE statement that specified either the PACK or NOPACK keyword, it was found that a trace was already active. The TRACE statement keywords, PACK and NOPACK, may not be specified while a trace is active.

PTE10547 - TRACE TO DDNAME IS NOT VALID

While executing a PSL TRACE statement that specified the TO keyword, it was found that the value following the TO keyword was not a 1 through 8 character actual, literal or variable ddname.

PTE10548 - RECORD PACK/NOPACK IS INVALID WHILE RECORD IS ACTIVE

While executing a PSL RECORD statement that specified either the PACK or NOPACK keyword, it was found that a recording was already active. The RECORD statement keywords, PACK and NOPACK, may not be specified while a recording is active.

PTE10550 - RECORD TO DDNAME IS NOT VALID

While executing a PSL RECORD statement that specified the TO keyword, it was found that the value following the TO keyword was not a 1 through 8 character actual, literal or variable ddname.

PTE10551 - RECORD TO DDNAME IS NOT VALID

While parsing a PSL RECORD statement that specified the TO keyword, it was found that the value following the TO keyword was not a 1 through 8 character actual, literal or variable ddname.

PTE10552 - TRACE TO DDNAME IS NOT VALID

While parsing a PSL TRACE statement that specified the TO keyword, it was found that the value following the TO keyword was not a 1 through 8 character actual, literal or variable ddname.

PTE10553 - A LABEL OR WORD MAY NOT EXCEED 71 CHARACTERS

While parsing a script statement, a label or other word was found that contained more than 71 characters. Because only statement labels may begin in position one of a script source statement, and because other script statement elements must be contained within columns 2 through 72 of a script source statement, labels in excess of 71 characters can not be used in GOTO, PERFORM or other statements and are not permitted.

PTE10554 - A LITERAL VALUE MAY NOT EXCEED 69 CHARACTERS

While parsing a script statement constructed by the EXECUTE statement, a literal value was found that contained more than 69 characters. A literal value contained in a non-executed statement can not exceed 69 characters because the literal can not begin in source statement position one, and including the quotes, the literal must fit in positions 2 through 72 of an 80 character script statement. The restriction applies to script statements constructed by the EXECUTE statement. Values up to 32760 bytes in length may be constructed by using the SET statement to concatenate literal values.

PTE10555 - A VARIABLE NAME MAY NOT EXCEED 70 CHARACTERS

While parsing a script statement constructed by the EXECUTE statement, a variable name was found that contained more than 70 characters. A variable name contained in a non-executed statement can not exceed 70 characters because the name can not begin in source statement position one, and including the ampersand that identifies variable names, the name must fit in positions 2 through 72 of an 80 character script statement. The restriction applies to script statements constructed by the EXECUTE statement.

PTE10556 - REDUNDANT OR CONFLICTING SET KEYWORD FOUND

While parsing a SET statement, one of the keywords, PRECISION, SIGNED, SIGN, PUNCTUATE or JUSTIFY was found twice, or two of the SET adjectives, ALL, DECIMAL, ZONED, PACKED, BINARY, STRING, LENGTH, HEX, IMAGE BITS, CAPITAL, or ATTRIBUTES, were found in a single SET statement.

PTE10557 - VALUE AFTER SET PRECISION KEYWORD NOT 0-30

The PRECISION keyword was specified in a PSL SET statement however the value following the PRECISION keyword was not an integer from zero through thirty. CA Enterprise Tester supports numeric integers and numbers with a maximum of 30 decimal places.

PTE10558 - USE PRECISION FOR BINARY/PACKED/ZONED ONLY

The PSL SET statement PRECISION keyword is used to specify the number of decimal places to be retained when a number is converted from decimal to BINARY, PACKED or ZONED format, or to specify the number of decimal places in a BINARY, PACKED or ZONED number when the number is expressed in decimal format. The PRECISION keyword was specified in a SET statement that did not also specify the BINARY, ZONED or PACKED keyword.

PTE10559 - USE SIGNED WITH TO/FROM BINARY ONLY

The PSL SET statement SIGNED keyword is used to specify whether or not the high order bit of a BINARY number represents an algebraic sign. The SIGNED keyword was specified in a SET statement that did not also specify the BINARY keyword. Use the SIGNED keyword in a SET statement only when converting a BINARY data item to decimal if the high order bit of the binary number represents a sign, or when converting a decimal number to binary to specify that the high order bit of the resulting binary number is to be used as a sign.

PTE10560 - VALUE AFTER JUSTIFY NOT LEFT/CENTER/RIGHT

The PSL SET statement JUSTIFY keyword is used to specify how a number is to be aligned in specified target field. One of the keywords, LEFT, CENTER or RIGHT must be specified immediately following the JUSTIFY keyword. LEFT specifies that the sign and decimal value is to be stored in the leftmost part of the specified target field. CENTER specifies that the sign of a number is to be left justified in the target field and that the numeric part of the decimal number is to be right justified in the target field. RIGHT specifies that the sign and decimal value is to be stored in the rightmost part of the specified target field.

PTE10561 - UP/DOWN INCREMENT/DECREMENT NOT DECIMAL

The source data item specified following the UP or DOWN keyword in a PSL SET statement did not contain a valid decimal value. Valid decimal numbers may contain a leading sign, an integer value including commas, a decimal point and a decimal fraction. The integer may not exceed 30 significant digits.

PTE10562 - COMPUTE PROCESSING TERMINATED

A PSL COMPUTE statement, listed following this message, was found to be invalid and was not executed. A message describing the COMPUTE statement error was listed preceding this message.

PTE10563 - LOGIC ERROR DETECTED WHILE EVALUATING EXPRESSION

A CA Enterprise Tester internal logic error was detected while evaluating an algebraic expression specified in a PSL COMPUTE statement or other statement that supports algebraic expressions. Report the error to CA Enterprise Tester technical support to obtain a program correction.

PTE10564 - 30 DIGIT INTEGER OVERFLOWED WHILE EVALUTATING EXPRESSION

While evaluating an algebraic expression specified in a PSL COMPUTE statement or other statement that supports algebraic expressions, an intermediate or final result exceeded 30 significant integer digits or if the statement specified the number of decimal places to be retained using the ROUND keyword, significant high order integer digits would be lost if the intermediate or final result were rounded to 30 digits including the specified number of digits in the decimal fraction. CA Enterprise Tester supports decimal numbers that contain a maximum of 30 decimal digits.

PTE10565 - NUMBER INVALID OR TOO BIG: "value"

While evaluating an algebraic expression specified in a PSL COMPUTE statement or other statement that supports algebraic expressions, the value contained in a variable field, screen field or other variable data item was found that did not contain a valid decimal value or that contained a decimal value that exceeded 30 significant integer digits. A valid decimal number may contain a leading sign, an integer value including commas, a decimal point and a decimal fraction. The integer may not exceed 30 significant digits.

PTE10566 - UNABLE TO DIVIDE BY ZERO

While performing a divide operation contained in an algebraic expression specified in a PSL COMPUTE statement or other statement that supports algebraic expressions, a divisor whose value was zero was encountered. CA Enterprise Tester does not support division by zero.

PTE10567 - 30 DIGIT INTEGER OVERFLOWED IN DIVIDE

While performing a divide operation contained in an algebraic expression specified in a PSL COMPUTE statement or other statement that supports algebraic expressions, a quotient was computed that exceeded 30 significant integer digits. CA Enterprise Tester supports decimal numbers with a maximum of 30 significant integer digits.

PTE10568 - SET SOURCE DATA IS NOT DECIMAL

The source data item specified in a PSL SET statement that specified the DECIMAL adjective or that used one of the adverbs SIGN, PUNCTUATE or JUSTIFY to edit decimal source data, did not contain a valid decimal number. Valid decimal numbers may contain a leading sign, an integer value including commas, a decimal point and a decimal fraction. The integer may not exceed 30 significant digits.

PTE10569 - ZONED TARGET SUBSTRING MAY NOT EXCEED 30 DIGITS

A PSL SET statement used to convert decimal data TO ZONED format specified a target substring length that exceeded 30 digits (bytes). CA Enterprise Tester support for decimal and zoned decimal numbers is limited to 30 digit numbers.

PTE10570 - PACKED SOURCE DATA EXCEEDS 30 DIGITS

The value of the packed source data item specified in a PSL SET statement that specified the FROM PACKED option exceeded 30 significant decimal digits. CA Enterprise Tester support for decimal and packed decimal numbers is limited to 30 digit numbers.

PTE10571 - TO BINARY SOURCE VALUE < 0: SPECIFY SIGNED

The value of the binary source data item specified in a PSL SET statement that specified the TO BINARY option was a negative value, however, the SET statement did not also specify the SIGNED adjective. By default, CA Enterprise Tester treats all bits in a binary number as part of the numeric value. When signed values are converted to binary, a SET statement must specify "TO SIGNED BINARY" to instruct CA Enterprise Tester to use the high order bit of the resulting binary number as a sign.

PTE10572 - BINARY TARGET MAY NOT EXCEED 8 BYTES

The substring length of the target field specified in a CA Enterprise Tester PSL SET statement that also specified the TO BINARY option was greater than eight bytes. CA Enterprise Tester supports signed and unsigned binary numbers from one through eight bytes only.

PTE10573 - SET SOURCE VALUE EXCEEDS TARGET SUBSTRING LENGTH

A PSL SET statement specified a target field substring length that was too short to contain the edited decimal source data. When a target substring length is specified insure that the length is large enough to contain the edited source value including any sign, commas, decimal point and significant numeric digits contained in the source value or implied by SIGN, PUNCTUATE and/or PRECISION keywords.

PTE10574 - SET TARGET FIELD WOULD EXCEED 32760

A PSL SET statement specified a target field substring position and source data that would result in a target variable whose length would exceed 32760 bytes of data if the source data were positioned in the variable field at the specified target field position. The maximum length of a CA Enterprise Tester variable field is 32760 bytes.

PTE10575 - TARGET VARIABLE NOT FOUND FOLLOWING COMPUTE

While parsing a PSL COMPUTE statement, an invalid expression was found following the COMPUTE verb. The value specified immediately following a COMPUTE verb must be a variable field expression that specifies the variable field or occurrence of a variable field array in which the computed result is to be stored. The value found was not a valid variable field expression.

PTE10576 - "PTE" VARIABLE MAY NOT BE ALTERED USING COMPUTE

The result field, specified immediately following a PSL COMPUTE verb was a variable field whose name began with the characters 'PTE'. Variable fields that begin with 'PTE' are defined by CA Enterprise Tester and are used to provide specific information to a script. The COMPUTE statement may not be used to alter the value of variable fields whose names begin with the characters 'PTE'.

PTE10577 - "=" NOT FOUND FOLLOWING COMPUTE VARIABLE

While parsing a PSL COMPUTE statement, a value other than an equal sign, '=', was found following the variable name in which the computed result was to be stored. In a COMPUTE statement, an equal sign must be coded immediately following the variable field name in which the computed result is to be stored and immediately preceding the algebraic expression to be evaluated.

PTE10578 - SYMBOL "x" NOT VALID IN EXPRESSION

An unsupported symbol was found within an algebraic expression specified in a PSL COMPUTE statement or other statement that supports algebraic expressions. When the message is issued, 'x' is replaced by the unsupported symbol that was encountered. Algebraic expressions may use the unary or arithmetic symbols, +, -, *. /, or may use parenthesis, (and), to specify the order in which arithmetic operations are to be performed. Other special symbols are not defined and are not supported.

PTE10579 - "*" OR "/" DOES NOT FOLLOW DATA, ROUND OR ")"

A multiplication symbol, '*', or a division symbol, '/' was found within an algebraic expression specified in a PSL COMPUTE statement or other statement that supports algebraic expressions, however, the symbol was not in a logical order relative to the data items to be multiplied or divided. Specifically, a multiplication or division symbol may be coded only following a data item or immediately following a right parenthesis that is used to group a previous inner expression whose result is to be used as the first data item in a multiply or divide operation.

PTE10580 - "(" DOES NOT FOLLOW "(", OR ARITHMETIC OPERATOR

A left parenthesis, '(', was found within an algebraic expression specified in a PSL COMPUTE statement or other statement that supports algebraic expressions, however, the symbol was not in a logical order relative to other parts of the algebraic expression. Parenthesis may be used in any logical way to order arithmetic operations in an algebraic expression. Specifically, left parenthesis may be used to begin inner expressions that are to be evaluated first and may be coded at the beginning of an expression, may be coded following another left parenthesis or may be coded following an arithmetic operation to begin an inner expression whose result is the data item to be used in the previous operation. Left parenthesis may not be coded at other points within an algebraic expression.

PTE10581 - ")" DOES NOT FOLLOW ")", DATA ITEM OR ROUND

A right parenthesis, ')', was found within an algebraic expression specified in a PSL COMPUTE statement or other statement that supports algebraic expressions, however, the symbol was not in a logical order relative to other parts of the algebraic expression. Parenthesis may be used in any logical way to order arithmetic operations in an algebraic expression. Specifically, right parenthesis are used to end inner expressions begin by a left parenthesis and may be coded following another right parenthesis, following the last data item of an inner arithmetic operation or following a rounding operation for a preceding inner expression. Right parenthesis may not be coded at other points within an algebraic expression.

PTE10582 - ")" IS NOT MATCHED BY PREVIOUS "("

A right parenthesis, ')' was found within an algebraic expression specified in a PSL COMPUTE statement or other statement that supports algebraic expressions, however, the right parenthesis is not matched by a previous left parenthesis, '(', that began an inner expression. Parenthesis may be used in any logical way to order arithmetic operations in an algebraic expression. Inner expressions contained within matching left and right parenthesis are evaluated first. Each inner expression must begin with a left parenthesis and end with a right parentheses. Inner expressions may contain inner expressions that are contained within additional pairs of left and right parenthesis.

PTE10583 - ROUND AFTER DATA OR ")" ONLY

The ROUND keyword was found within an algebraic expression specified in a PSL COMPUTE statement or other statement that supports algebraic expressions, however, the ROUND keyword did not follow a data item or a right parenthesis that was used to close an inner expression. A rounding operation may be performed only on a previous data item, the result of a previous arithmetic operation or the result of a previous inner expression and must immediately follow the data item, operands of the arithmetic operation or the closing parenthesis of a previous inner expression.

PTE10584 - UP, DOWN, OR 0-30 NOT FOUND FOLLOWING ROUND

The ROUND keyword was found within an algebraic expression specified in a PSL COMPUTE statement or other statement that supports algebraic expressions, however required operands of the round operation were not found following the ROUND keyword. Following the ROUND keyword one of the keywords, UP or DOWN, or an integer from 0 through 30, specifying the number of decimal places to which the result is to be rounded, must be coded. If UP or DOWN are specified, the number of decimal places must be coded following the UP or DOWN keyword.

PTE10585 - ABSOLUTE NOT AFTER "(" OR ARITHMETIC OPERATOR

The ABSOLUTE keyword was found within an algebraic expression specified in a PSL COMPUTE statement or other statement that supports algebraic expressions, however the keyword was not in a logical order relative to other elements of the expression. The ABSOLUTE keyword must precede the data item or inner expression, the absolute value of which is desired. Specifically, the ABSOLUTE keyword may be coded as the first element of an expression, the first element of an inner expression, or may be coded following an arithmetic operator that precedes a subsequent data item or inner expression.

PTE10586 - DATA ITEM DOES NOT FOLLOW "(" OR ARITHMETIC OPERATOR

A data item was found within an algebraic expression specified in a PSL COMPUTE statement or other statement that supports algebraic expressions, however the data item was not in a logical order relative to other elements of the expression. Specifically, a data item may appear in an expression as the first element of an expression, as the first element of an inner expression or following an arithmetic operator including the ABSOLUTE value function or a unary operator.

PTE10587 - EXPRESSION SCRATCH PAD EXCEEDS 32760 BYTES

A PSL COMPUTE statement or other statement that supports the evaluation of algebraic expressions was found that required an intermediate result work area, referred to as an 'expression scratch pad', that exceeded 32760 bytes of storage. The size of the area required is a function of the number of intermediate results that must be calculated in order to evaluate the entire expression. This error is detected only while parsing expressions of extraordinary complexity. If you have encountered this error, it is suggested that you break the expression down into multiple COMPUTE statements. If your expression is not complex, contact CA Enterprise Tester Technical Support for assistance in determining the cause of the error.

PTE10588 - EXPRESSION CONTAINS INCOMPLETE OPERATION OR MISSING ")"

A PSL COMPUTE statement or other statement that evaluates algebraic expressions was parsed, however the end of the expression was incomplete. Either a data item required by a previously specified arithmetic or unary operation is missing or one or more right parenthesis are required to close specified inner expressions.

PTE10589 - NUMBER INVALID OR TOO BIG

A literal value was found within an algebraic expression specified in a PSL COMPUTE statement or other statement that evaluates algebraic expressions.

While literal values are permitted to allow punctuation of integers in script text, the literal must contain a single valid decimal value that does not exceed 30 significant integer digits. The literal value specified was either not a valid decimal value or was a number that exceeded 30 significant integer digits.

PTE10590 - SET TARGET FOR UP/DOWN MAY NOT BE A SUBSTRING

A variable substring was specified in a PSL SET UP or DOWN statement as the variable field to be incremented or decremented by some value. The variable specified immediately following the SET verb must be a variable field or occurrence in a variable field array when the UP or DOWN keyword is specified. A substring of a variable field may not be specified as the target field when UP or DOWN is specified. Another SET statement that does not specify UP or DOWN may be used to edit and/or justify an incremented or decremented value or other numeric values into a desired substring of a variable field.

PTE10591 - TARGET FIELD LENGTH REQUIRED FOR JUSTIFY

A PSL SET statement that specified the JUSTIFY keyword to justify data in the left part, center of, or right part of a target variable field substring, did not also specify the length of the target field substring. When JUSTIFY is specified the target variable field expression must specify a substring length in order to specify the length of the area in which the source data is to be justified.

PTE10592 - IS/NOT DOES NOT PRECEED A NUMERIC TEST KEYWORD

One of the keywords, IS or NOT, was found in a relational expression contained in a PSL IF or DO statement, however the value following the keyword was not one of the numeric test keywords, DECIMAL, ZONED, or PACKED. "IF &V IS DECIMAL" (etc.) is true if &V contains a valid decimal value. "IF &V IS NOT DECIMAL" (etc.) is true if &V does not contain a valid decimal value. Relational expressions may specify the NOT keyword preceding numeric test keywords only. The keyword, IS, is optional.

PTE10593 - "*" MAY NOT BE USED AS A LENGTH FOR NUMERIC TESTS

A variable field or screen field expression specified in a relational expression contained in a PSL IF or DO statement specified the special symbol, '*', as a substring or screen field length. In a relational expression, a length of '*', normally means to use the length of the other argument in the relational expression as the length of the first argument. However, the relational condition was one of the numeric test keywords, DECIMAL, ZONED or PACKED, and a second data argument is not valid. Therefore, there is no other source from which to obtain the length of the first argument. As a result, a substring of '*' may not be used in a relational expression that uses the DECIMAL, ZONED or PACKED condition to test the numeric properties of a data item.

PTE10594 - MINUS SIGN NOT FOLLOWED BY A NUMBER

A minus sign, '-', was encountered while parsing a PSL statement, indicating that a number should follow and that the negative of the number is the desired value of the data item. However, a number was not found following the minus sign. When a negative number is coded by coding the minus sign, specify the number immediately following the minus sign.

PTE10595 - COMPUTE TARGET MAY NOT BE A SUBSTRING

A variable substring was specified as the result field of a PSL COMPUTE statement. The result field specified immediately following the COMPUTE verb must be a variable field or occurrence in a variable field array. A substring of a variable field may not be specified. The SET statement, which also supports editing and justification of computed values and other numbers, may be used to place a computed value into a desired substring of another variable field.

PTE10596 - PTETRACE DD OPEN FAILED, SCRIPT NOT EXECUTED

A CA Enterprise Tester script was executed in a batch job using CA Enterprise Tester program, PTEBS. A PTETRACE DD statement was present, indicating that any session created by the script was to be traced. However, the target data set specified by the PTETRACE DD statement was not a suitable target for a CA Enterprise Tester trace. A trace may be written to a data set, PDS member, or SYSOUT provided that the target is RECFM=FBA and LRECL=135. A message that precedes this message states the problem with the target specified on the PTETRACE DD statement. After this message is issued, the job step is terminated with a user 4095 abend. Specify a valid target data set or remove the PTETRACE DD statement if a session trace is not desired.

PTE10597 - DROP UNQUALIFIED NON-PTE VARIABLE NAME only

A PSL DROP statement specified a variable name that begins with the characters, 'PTE', or specified a variable subscript, substring position or substring length. The DROP statement may not be used to drop CA Enterprise Tester defined variables. The DROP statement returns a user defined variable field or variable field array to its initial non-existent state and may not be used to operate on a substring of a variable field or an occurrence in a variable field array. Specify only the variable field name or names to be dropped.

PTE10598 - VARIABLE NAME NOT FOUND FOLLOWING DROP

A PSL DROP statement was found, but a variable field name was not found following the DROP verb. The DROP statement returns a user defined variable field or variable field array to its initial non-existent state and may not be used to operate on a substring of a variable field or an occurrence in a variable field array. When a DROP statement is coded, specify at least one variable field name.

PTE10599 - REDUNDANT OR CONFLICTING COMPUTE KEYWORD FOUND

While parsing a COMPUTE statement, one of the keywords, SIGN, PUNCTUATE or JUSTIFY was found twice. Do not code redundant COMPUTE keywords.

PTE10600 - VALUE AFTER JUSTIFY NOT LEFT/CENTER/RIGHT

The PSL COMPUTE statement JUSTIFY keyword is used to specify how a number is to be aligned in specified target field. One of the keywords, LEFT, CENTER or RIGHT must be specified immediately following the JUSTIFY keyword. LEFT specifies that the sign and decimal value is to be stored in the leftmost part of the specified target field. CENTER specifies that the sign of a number is to be left justified in the target field and that the numeric part of the decimal number is to be right justified in the target field. RIGHT specifies that the sign and decimal value is to be stored in the rightmost part of the specified target field.

PTE10601 - TARGET FIELD LENGTH REQUIRED FOR JUSTIFY

A PSL COMPUTE statement that specified the JUSTIFY keyword to justify data in the left part, center of, or right part of a target variable field substring, did not also specify the length of the target field substring. When JUSTIFY is specified the target variable field expression must specify a substring length in order to specify the length of the area in which the source data is to be justified.

PTE10602 - SUBSTRING POSITION + LENGTH GT 32760

A PSL COMPUTE statement specified that a substring of a variable field was to be set to some source value. The sum of the target field substring position plus the length of the specified source data exceeded the maximum length of a single CA Enterprise Tester variable field.

PTE10603 - THINK TIME NOT HHMMSSTH

The time specified following a PSL THINK verb was not a string of numeric digits of the form, HHMMSSTH, expressing time in hours, minutes, seconds, tenths and hundredths of seconds. Leading zero digits need not be specified. No more than 8 digits may be specified.

PTE10604 - THINK RATE NOT VALID NUMBER OR VARIABLE

An invalid data item was found following a PSL THINK statement RATE keyword. The data item following RATE may be a variable field expression or may be a number greater than zero.

PTE10605 - THINK PER TIME NOT HHMMSSTH OR VARIABLE

An invalid data item was found following a PSL THINK statement PER keyword. PER time may be expressed as a variable field expression or may be a string of numeric digits of the form, HHMMSSTH, expressing time in hours, minutes, seconds, tenths and hundredths of seconds. Some amount of time must be specified (0 is invalid). Leading zero digits need not be specified. No more than 8 digits may be specified.

PTE10606 - THINK SESSIONS NOT VALID NUMBER OR VARIABLE

An invalid data item was found following a PSL THINK statement SESSIONS keyword. The data item following SESSIONS may be a variable field expression or may be an integer greater than zero.

PTE10607 - THINK SAMPLE NOT VALID NUMBER OR VARIABLE

An invalid data item was found following a PSL THINK statement SAMPLE keyword. The data item following SAMPLE may be a variable field expression or may be an integer from 1 to 1000.

PTE10608 - THINK RTIME NOT HHMMSSTH OR VARIABLE

An invalid data item was found following a PSL THINK statement RTIME keyword. The RTIME value may be expressed as a variable field expression or may be a string of numeric digits of the form, HHMMSSTH, expressing time in hours, minutes, seconds, tenths and hundredths of seconds. Leading zero digits need not be specified. No more than 8 digits may be specified.

PTE10609 - REDUNDANT THINK RATE KEYWORD FOUND

A PSL THINK statement specified one of the keywords, RATE, PER, SESSIONS, SAMPLE or RTIME more than one time. Specify THINK statement keywords only one time per THINK statement.

PTE10610 - THINK RATE REQUIRED WITH KEYWORD CODED

A PSL THINK statement specified one of the keywords PER, SESSIONS, SAMPLE or RTIME, but the RATE keyword was not specified. PER, SESSIONS, SAMPLE and RTIME keywords that influence the session rate and RATE must be specified when other THINK keywords are specified.

PTE10611 - THINK TIME NOT HHMMSSTH

The variable field specified following a PSL THINK verb was not a character variable field or did not contain a valid time value. A valid time value is a string of numeric digits of the form, HHMMSSTH, expressing time in hours, minutes, seconds, tenths and hundredths of seconds. Leading zero digits need not be specified. No more than 8 digits may be specified.

PTE10612 - THINK RATE NOT A POSITIVE NUMBER

The variable field specified following a PSL THINK statement RATE keyword contained an invalid value. The RATE value must be a number greater than zero.

PTE10613 - THINK PER TIME IS 0 OR NOT HHMMSSTH

The variable field specified following a PSL THINK statement PER keyword was not a character variable field or did not contain a valid, non-zero, time value. A valid time value is a string of numeric digits of the form, HHMMSSTH, expressing time in hours, minutes, seconds, tenths and hundredths of seconds. Leading zero digits need not be specified. No more than 8 digits may be specified.

PTE10614 - THINK SESSIONS NOT POSITIVE INTEGER

The variable field specified following a PSL THINK statement SESSIONS keyword contained an invalid value. The SESSIONS value must be an integer greater than zero.

PTE10615 - THINK SAMPLE NOT 1 THROUGH 1000

The variable field specified following a PSL THINK statement SAMPLE keyword contained an invalid value. The SAMPLE value must be an integer from 1 to 1000.

PTE10616 - THINK RTIME NOT HHMMSSTH

The variable field specified following a PSL THINK statement RTIME keyword was not a character variable field or did not contain a valid time value. A valid time value is a string of numeric digits of the form, HHMMSSTH, expressing time in hours, minutes, seconds, tenths and hundredths of seconds. Leading zero digits need not be specified. No more than 8 digits may be specified.

PTE10617 - DO NOT SPECIFY WHILE/UNTIL FOLLOWING FOREVER

While parsing a DO statement, the FOREVER keyword was found, however one of the verbs, WHILE or UNTIL was found following FOREVER. Specify only one of the keywords FOREVER, WHILE or UNTIL.

PTE10618 - SPECIFY WHILE/UNTIL FOLLOWING DO BEFORE/AFTER

While parsing a DO statement, one of the keywords BEFORE or AFTER was found, however one of the keywords WHILE or UNTIL was not found following BEFORE or AFTER. When BEFORE or AFTER are specified, WHILE or UNTIL must also be specified.

PTE10619 - ITERATE NOT WITHIN A DO ROUTINE

While parsing a script, prior to script execution, an ITERATE statement was encountered. The ITERATE statement is invalid because it is not within a DO routine. Use the ITERATE statement only within a DO routine.

PTE10620 - LEAVE NOT WITHIN A DO ROUTINE

While parsing a script, prior to script execution, a LEAVE statement was encountered. The LEAVE statement is invalid because it is not within a DO routine. Use the LEAVE statement only within a DO routine.

PTE10621 - SPECIFY PASS ONLY WITH RECORD TO

While parsing a script, prior to script execution, a RECORD statement was encountered that specified the PASS keyword, however the RECORD statement did not also specify the TO keyword. PASS may be specified only when RECORD TO is specified.

PTE10622 - SPECIFY PASS ONLY WITH TRACE TO

While parsing a script, prior to script execution, a TRACE statement was encountered that specified the PASS keyword, however the TRACE statement did not also specify the TO keyword. PASS may be specified only when TRACE TO is specified.

PTE10623 - SELECTED SCREEN LOCATION IS NOT PEN DETECTABLE

A PSL TYPE statement SELECT keyword was executed while the virtual terminal cursor was not positioned in a pen detectable field with a valid designator character in position 1 of the field. To perform operations on a pen detectable field similar to those performed by the cursor select key or a light pen, position the cursor somewhere within a pen detectable field before executing the TYPE SELECT keyword. The pen detectable field may be either a selection or pen attention field.

PTE10624 - DESELECTED SCREEN LOCATION IS NOT PEN DETECTABLE

A PSL TYPE statement DESELECT keyword was executed while the virtual terminal cursor was not positioned in a pen selectable field with a valid field designator character, ? or >. To deselect a pen selectable field, position the cursor somewhere within the selectable field before executing the TYPE DESELECT keyword.

PTE10625 - DELAY STATEMENT LOGIC ERROR

An unknown logic error was detected while executing a PSL DELAY statement. If you receive this message, please take note of the circumstances leading up to the DELAY statement error and contact CA Enterprise Tester technical support for assistance.

PTE10626 - DELAY TIME NOT HHMMSSTH

A PSL DELAY statement was executed that specified a variable field as delay time. However, during execution of the statement, it was found that the variable field value was not a valid delay time expressed in the form, hhmmssth, where hh is hours, mm is minutes, ss is seconds, and th is tenths and hundredths of a second.

PTE10627 - DELAY PERCENT NOT 0 THROUGH 999',VE/PTBCSXCV

A PSL DELAY statement was executed that specified the PERCENT keyword and a variable field as the delay percent. However, during execution of the statement, it was found that the variable field did not contain a valid percent. A valid DELAY PERCENT is an integer from 0 through 999. and may be expressed Specify a percent from 0 through 99 to reduce DELAY statement times and make a script session run faster, or specify a percent from 101 to 999 to extend DELAY statement times and to make a script session run slower.

PTE10628 - OPEN UNPACK DATASET NOT RECFM=F OR LRECL>256

While executing a PSL OPEN statement that specified the UNPACK keyword, it was found that the dataset opened was not fixed length records or contained logical records greater than 256 bytes. The UNPACK option may be specified to automatically unpack and return logical records when requested by subsequent PSL READ statements, however UNPACK support is limited to datasets with fixed-length logical records up to 256 bytes in length.

PTE10629 - TARGET FOR DDNAME RETURN INVALID

While executing a PSL ALLOCATE statement that requested the return of a system assigned ddname by specifying ddname as a variable field that contained no value, it was found that the variable ddname return field expression also specified a substring length less than8 characters. System assigned ddnames are 8 characters in length and can not be returned in a variable field substring whose length is less than 8.

PTE10630 - RECORD COMMENTS NOT 0 THROUGH 43

While executing a PSL RECORD statement that specified the COMMENTS keyword and a variable number of screen comments to be recorded, it was found that the value of the variable field was not an integer from 0 through 43. To suppress the recording of screen comments, specify COMMENTS 0 or a variable field that contains 0. To record a desired number of screen comment lines, specify the number of lines, 1-43, or a variable field that contains 1-43.

PTE10631 - RECORD LANGUAGE INVALID WHILE RECORDING IS ACTIVE

While executing a PSL RECORD statement that specified either the PSL or REXX keyword, it was found that a session recording was already active. The PSL and REXX keywords may be used to specify that a recorded session be recorded in either the CA Enterprise Tester Script Language, by specifying PSL, or that a session is to be recorded as a REXX exec, by specifying REXX. However, the language can not be changed while a recording is active, and therefore, the PSL or REXX keywords may be specified only on RECORD statements that are executed while a recording is not already active.

PTE10632 - TRACK STATEMENT LABEL COLUMN INVALID: trace-line

While executing a PSL TRACK statement that specified the STATEMENT keyword, a non-screen, non-title trace dataset record was found that contained other than a blank or a comment delimiter, '*', /* or */, in position 2 of the trace dataset record. The record was not within a series of records that constitute a single /* */ comment. When data other than captured screen images, titles and PSL statements are inserted in a CA Enterprise Tester trace data set, record position 1 is reserved as a RECFM=A control character and record positions 2 through 73 are seen as script statement data, normally positions 1 through 72 of PSL script records. When TRACK STATEMENT is used to read a trace dataset, record positions 2-73 of records inserted into a trace dataset must conform to CA Enterprise Tester Script Language rules, except that script labels are not supported for script statements contained in a trace dataset. Therefore only comment delimiters and comment data within a multi-record /* */ comment may exist in position 2 of trace dataset PSL records that are to be processed using a TRACK statement that specifies the STATEMENT keyword. When the message is issued, 'trace-line', is replaced by the invalid trace record.

PTE10633 - TRACK STATEMENT VERB NOT FOUND: trace-line

While executing a PSL TRACK statement that specified the STATEMENT keyword, a non-screen, non-title trace dataset record was found that contained was not a PSL verb and was not contained within a PSL statement. When TRACK STATEMENT is used to read a trace dataset, record positions 2-73 of records inserted into a trace dataset must conform to CA Enterprise Tester Script Language rules, and must contain statements that begin with a PSL verb. When the message is issued, 'trace-line', is replaced by the invalid trace record.

PTE10634 - TRACK STATEMENT CLOSE QUOTE NOT FOUND: trace-line

While executing a PSL TRACK statement that specified the STATEMENT keyword, a PSL statement was being parsed that contained a literal value, however, the closing quote was not found within record positions 3-73 of the trace record. When TRACK STATEMENT is used to read a trace dataset, record positions 2-73 of records inserted into a trace dataset must conform to CA Enterprise Tester Script Language rules, and any literal value contained within a PSL statement must be wholly contained within record positions 3-73, including both opening and closing quotes. When the message is issued, 'trace-line', is replaced by the invalid trace record.

PTE10635 - TRACK STATEMENT /* DLM */ NOT FOUND

While executing a PSL TRACK statement that specified the STATEMENT keyword, the trace dataset end-of-file was reached while parsing a /* comment that was not completed by an */ comment delimiter. When TRACK STATEMENT is used to read a trace dataset, record positions 2-73 of records inserted into a trace dataset must conform to CA Enterprise Tester Script Language rules, and any /* comment must be delimited by a */ comment delimiter on the same or a subsequent trace record, to end the comment.

PTE10636 - TRACK DATA SET CONTAINS NON-FORMAT SCREEN DATA

While executing a PSL TRACK statement, data was found that was not captured using a CA Enterprise Tester session trace that specified the Format option. The TRACK statement may be used only to process a CA Enterprise Tester trace that was created using the Format option and only when the data set LRECL is at least 135 characters. The data set specified does not contain screen images that can be processed by the TRACK statement.

PTE10637 - PTEIN DATASET NOT RECFM=F LRECL=80

The PTEIN DD statement of a job step that executed PGM=PTEBS, the CA Enterprise Tester Script Language processor, contained other than fixed-length, 80-character records. CA Enterprise Tester scripts consist of fixed-length 80-character records and may not be executed from a dataset that conflicts with the PSL record format requirements.

PTE10638 - LINK dsn(mem) FAILED - PTELIB ALLOC ERROR

While attempting to load a CA Enterprise Tester script specified by a PSL LINK statement, it was found that no PTELIB DD was allocated or that the linked to script was not in any allocated PTELIB dataset. An attempt to locate the linked to script by dynamically concatenating any library referred to by the PTEIN DD statement and the CA Enterprise Tester system link library, to any allocated PTELIB datasets, was unsuccessful because dataset allocation failed. A message issued prior to this message documents the specific allocation failure.

PTE10639 - LINK dsn(mem) FAILED - PACKED DATA ERROR

An attempt to load a CA Enterprise Tester script specified by a PSL LINK statement was unsuccessful. The script member was found to be in the ISPF packed data format, however the script could not be unpacked because packed data control characters were missing or invalid. A message issued prior to this message documents the specific pack data control data found to be invalid . If the cause of the failure can not be determined, contact CA Enterprise Tester technical support for assistance.

PTE10640 - LINK dsn FAILED - DCB NOT RECFM=F LRECL=80

An attempt to load a CA Enterprise Tester script specified by a PSL LINK statement was unsuccessful because a dataset specified via the PTELIB DD statement of a job step that executed PGM=PTEBS, contained other than fixed-length 80-character records. CA Enterprise Tester scripts consist of fixed-length 80-character records and may not be executed from a dataset that conflicts with the PSL record format requirements.

PTE10641 - ALLOCATE FAILED FOR LIB=dsn, RC=xxxxxxxx

While attempting to load a CA Enterprise Tester script specified by a PSL LINK statement, it was found that no PTELIB DD was allocated or that the linked to script was not in any allocated PTELIB dataset. An attempt to locate the linked to script by dynamically concatenating any library referred to by the PTEIN DD statement and the CA Enterprise Tester system link library, to any allocated PTELIB datasets, was unsuccessful because dataset allocation failed. When the message is issued, 'xxxxxxxx' is replaced by the SVC 99 error reason and information code. Refer to Appendix A of the help that is available when the HELP key is pressed while viewing the ISPF/PDF PRIMARY OPTION MENU or other IBM documentation of SVC 99 services for an explanation of the dynamic allocation error reason and information code.

PTE10642 - PTELIB CONCATENATE FAILED, RC=xxxxxxxx

While attempting to load a CA Enterprise Tester script specified by a PSL LINK statement, it was found that no PTELIB DD was allocated or that the linked to script was not in any allocated PTELIB dataset. An attempt to locate the linked to script by dynamically concatenating any library referred to by the PTEIN DD statement and the CA Enterprise Tester system link library, to any allocated PTELIB datasets, was unsuccessful because dynamic concatenation of the selected libraries failed. When the message is issued, 'xxxxxxxx' is replaced by the SVC 99 error reason and information code. Refer to Appendix A of the help that is available when the HELP key is pressed while viewing the ISPF/PDF PRIMARY OPTION MENU or other IBM documentation of SVC 99 services for an explanation of the dynamic concatenation error reason and information code.

PTE10643 - DELAY MUST SPECIFY TIME OR PERCENT

While parsing a PSL DELAY statement, it was found that the statement specified neither a delay time nor the PERCENT keyword immediately following the DELAY keyword. A DELAY time may be expressed as an actual value in the form, hhmmssth, where hh is hours, mm is minutes, ss is seconds, and th is tenths and hundredths of a second, or may be a variable field that contains DELAY time in the form, hhmmssth. The first DELAY statement argument following the DELAY keyword must be either a valid delay time or the PERCENT keyword followed by a valid delay percent.

PTE10644 - DELAY PERCENT NOT A NUMBER OR VARIABLE

While parsing a PSL DELAY statement that specified the PERCENT keyword, it was found that the argument immediately following the PERCENT keyword was invalid. A DELAY PERCENT may be expressed as a decimal integer or as a variable field that contains a valid delay percent. Either specify an actual delay percent as an integer from 0 through 999 or specify a variable field that will contain a valid delay percent when the DELAY PERCENT statement is executed.

PTE10645 - LINK NOT EXECUTABLE

While parsing a PSL LINK statement that specified the name of a script to be linked to during execution of the LINK statement, an attempt was made to load the specified script. The script load failed, for a reason documented by a previous message. Because the script could not be successfully loaded, the LINK statement is not executable.

PTE10646 - RECORD COMMENTS NOT 0 THROUGH 43

While parsing a PSL RECORD statement that specified the COMMENTS keyword, however it was found that the value immediately following the RECORD keyword was not a valid number of screen comment lines to be recorded. The number of comment lines may be an integer from 0 through 43 or may be a variable field that will contain the number of screen comment lines to be recorded by the time the RECORD statement is executed. Either specify the actual number of screen comment lines to be recorded or specify a variable field and set the variable field to the number of screen comment lines to be recorded by the time that the RECORD statement is executed.

PTE10647 - TIMEOUT S522 OPTION NOT ON/OFF

While parsing a PSL TIMEOUT statement that specified the S522 keyword option, it was found that the value immediately following the S522 keyword was not ON or OFF. When TIMEOUT S522 is specified, specify either ON to allow S522 abends to occur when the job has waited on an event for an installation-specified time interval, or specify OFF to disable S522 abends and continue waiting until an event completes or until a specified TIMEOUT time has elapsed.

PTE10648 - SPECIFY TRACK LINE OR STATEMENT, NOT BOTH

While parsing a PSL TRACK statement, it was found that both of the TRACK statement keywords, LINE and STATEMENT was specified. The TRACK statement returns any non-screen image data contained in a CA Enterprise Tester trace data set as a line of data when LINE is specified, or returns complete PSL statements that may span multiple trace data set logical records when STATEMENT is specified. But it can not do both on the same TRACK statement. Specify either LINE or STATEMENT if you wish to process any non-screen data contained in a CA Enterprise Tester trace dataset using the TRACK statement.

PTE10649 - REDUNDANT OR CONFLICTING PAUSE OPTIONS FOUND

While parsing a PSL PAUSE statement, a keyword was found that was specified more than one time or conflicting keywords were specified. a keyword was found that is not supported by the WTO statement. A PSL PAUSE statement may specify each of the keywords, REPLY. AT, MESSAGE, and HELP, but each keyword may be specified only one time.

PTE10650 - PAUSE AT OPTION INVALID

While parsing a PSL PAUSE statement, a the AT keyword was found, but the name found following the AT keyword was not one of the supported keywords, SESSIONVIEW, SESSIONREVIEW, TRACEVIEW or TRACEREVIEW.

PTE10651 - PAUSE MESSAGE DATA NOT A LITERAL OR VARIABLE VALUE

While parsing a PSL PAUSE statement, the MESSAGE keyword was found, however the value found following the MESSAGE keyword was not a literal or variable data item.

PTE10652 - PAUSE HELP DATA NOT A LITERAL OR VARIABLE VALUE

While parsing a PSL PAUSE statement, the HELP keyword was found, however the value found following the HELP keyword was not a literal or variable data item.

PTE10653 - TRACK REVIEW/BACK/STACK N NOT 0 THROUGH 32,760

A PSL TRACK statement that specified that REVIEW< BACK, or STACK keyword specified a constant or variable value that was not an integer from 0 through 32.760. When TRACK REVIEW is used to set the minimum number TRACK responses to be reviewed, the limit is 32.760. When TREACK BACK is used to return previously tracked data, the value specified by the TRACK BACK statement must be a nonnegative integer from 0 to the current value of PSL variable PRETBACK. When TRACK BSTACK is used to return previously tracked data that was stacked by a previously executed TRACK BACK statement, the value specified by the TRACK ATAXK statement must be a non-negative integer from 0 to the current value of PSL variable PTETRACK.

PTE10654 - TRACK BACK N EXCEEDS PTETBACK VALUE

A PSL TRACK statement that specified the BACK keyword specified a constant or variable value that was greater than the current value of PSL variable, PTETBACK. The value of PTETBACK is the number of previously tracked screens, PSL statements or lines that have been saved for review, and is the maximum value that may be specified by a TRACK BACK statement. The minimum number of tracked screens saved for review may specified by a TRACK REVIEW statement that specifies a value from 0 to 32,760.

PTE10655 - TRACK STACK N EXCEEDS PTETRACK VALUE

A PSL TRACK statement that specified the STACK keyword, specified a constant or variable value that was greater than the current value of PSL variable, PTETRACK. The value of PTETRACK is the number of previously tracked screens, PSL statements or lines in the TRACK stack created when the TRACK BACK statement is used to return previously tracked data, and is the maximum value that may be specified by a TRACK STACK statement. Any TRACK stack created by TRACK BACK statements is automatically returned to the TRACK review queue when a TRACK statement that specifies a ddname is used to track data from a trace data set.

PTE20001 - UNKNOWN FUNCTION REQUESTED - ABENDING

The CA Enterprise Tester Access Method program, PTEAM, was CALLed by an application program. The application requested a service other than LOGON, CONVERSE, LOGOFF or WAIT. The job step is terminated with a user 4095 abend.

PTE20002 - LOGON REQUEST BUT SESSION ACTIVE - ABENDING

An application program CALLed the CA Enterprise Tester Access Method program, PTEAM. The application requested that the Access Method LOGON to an online system. An active session has already been obtained by the application via a previous LOGON CALL. If a session with a different online system is desired, the application must first terminate the active session via a LOGOFF CALL. The job step is terminated with a user 4095 abend.

PTE20003 - CONVERSE REQUEST BUT NO SESSION - ABENDING

An application program CALLed the CA Enterprise Tester Access Method program, PTEAM. The application specified a CONVERSE request. An active session has not previously been obtained via a LOGON CALL. A successful LOGON CALL must be completed prior to any CONVERSE CALL. The job step is terminated with a user 4095 abend.

PTE20004 - INVITE REQUEST BUT NO SESSION - ABENDING

The CA Enterprise Tester Access Method program, PTEAM, was CALLed by an application program. The application specified an INVITE request. An active session has not previously been obtained via a LOGON CALL. A successful LOGON CALL must be completed prior to any INVITE CALL. The job step is terminated with a user 4095 abend.

PTE30101 - INVALID FUNCTION CODE PASSED TO EXIT PROGRAM

The CA Enterprise Tester Communications Exit Program was called with a function code that requested other than LOGON, SIGNON, CONVERSE, or LOGOFF Communications Exit Program processing. Refer to the *CA Enterprise Tester System Guide* for details.

PTE30102 - PTXPAPPL PARM NOT SPECIFIED

The CA Enterprise Tester Communications Exit Program was called with a function code that requested LOGON Communications Exit Program processing. The address of an online system name was not specified. Refer to the *CA Enterprise Tester System Guide* for details.

PTE30103 - LOGON APPL ID REQUIRED

A CA Enterprise Tester Communications Exit Program was called with a function code that requested LOGON Communications Exit Program processing. The name of an online system was not specified.

PTE30104 - PTXPTERM PARM NOT SPECIFIED

A CA Enterprise Tester Communications Exit Program was called with a function code that requested LOGON Communications Exit Program processing. The address of a CA Enterprise Tester name was not specified. Refer to the *CA Enterprise Tester System Guide* for details.

PTE30105 - PTCC CB NOT FOUND AT SIGNOFF EXIT

A CA Enterprise Tester Communications Exit Program was called with a function code that requested SIGNOFF processing. The address of a CA Enterprise Tester control block associated with an active CA Enterprise Tester session was not provided. Refer to the *CA Enterprise Tester System Guide* for details.

PTE30106 - PTCC CB NOT FOUND AT SIGNON EXIT

A CA Enterprise Tester Communications Exit Program was called with a function code that requested SIGNON processing. The address of a CA Enterprise Tester control block associated with an active CA Enterprise Tester session was not provided. Refer to the *CA Enterprise Tester System Guide* for details.

PTE30107 - SYSTEM NOT DEFINED IN PTEOT

A CA Enterprise Tester Communications Exit Program was called to perform pre-logon processing. The logon request being processed specified an online system name that was not defined in the CA Enterprise Tester options table (PTEOT). An options table statement, PTEOM CONTROL,LOGON=ONLY, specified that only logons to systems defined by PTEOM SYSTEM statements are to be accepted. If the system name specified in the logon request is a valid system name that should be accessible via CA Enterprise Tester, then contact the systems programmer responsible for the installation and maintenance of CA Enterprise Tester and request that a PTEOM SYSTEM statement be added to the CA Enterprise Tester Options Table. to define the system specified in the rejected SESSION command or LOGON statement.

PTE30108 - PTCC CB NOT FOUND AT CONVERSE EXIT

A CA Enterprise Tester Communications Exit Program was called with a function code that requested CONVERSE processing. The address of a CA Enterprise Tester control block associated with an active CA Enterprise Tester session was not provided. Refer to the *CA Enterprise Tester System Guide* for details.

PTE30109 - PTXP NOT PASSED TO EXIT PROGRAM - ABENDING

A CA Enterprise Tester Communications Exit Program was called but no parameter was provided. A caller of the exit facility must provide parameters as described in the *CA Enterprise Tester System Guide* .

PTE30110 - PTBA NOT PASSED TO EXIT PROGRAM - ABENDING

A CA Enterprise Tester Communications Exit Program was called but the parameter area did not provide the address of a PTCC BASE area used by the Communications Exit Program to locate the CA Enterprise Tester session control block and to provide Communications Exit Program feedback information to the caller. Refer to the *CA Enterprise Tester System Guide* for parameters required by the Communications Exit Program.

PTE30111 - ALL PROGRAM TERMINALS IN THIS SYSTEM ARE BUSY

A CA Enterprise Tester Communications Exit Program was called to obtain a session with an online system. All of the terminals managed by the Communications Exit Program are currently in use. Specify a terminal id, or retry the LOGON request.

PTE30112 - THE NAMED PROGRAM TERMINAL IS BUSY

A CA Enterprise Tester Communications Exit Program was called to obtain a session with an online system. All of the terminals managed by the Communications Exit Program are currently in use. Specify a terminal id, or retry the LOGON request.

PTE30113 - PTXPLMOD PARM NOT SPECIFIED

A CA Enterprise Tester Communications Exit Program was called with a function code that requested LOGON Communications Exit Program processing. The address of a Logon Mode Table field was not supplied by the caller. Refer to the *CA Enterprise Tester System Guide* for details.

PTE40001 - TGET DATA RECEIVED FROM TERMINAL

The CA Enterprise Tester TLON command was entered at the Interactive System Control panel. TLON instructs CA Enterprise Tester to log the TPUT/TGET data sent or received by the CA Enterprise Tester Zoom Mode display manager. This message documents that the data area dump that follows is a data stream that was received by CA Enterprise Tester from the TSO terminal.

PTE40002 - TPUT DATA SENT TO TERMINAL

The CA Enterprise Tester TLON command was entered at the Interactive System Control panel. TLON instructs CA Enterprise Tester to log the TPUT/TGET data sent or received by the CA Enterprise Tester Zoom Mode display manager. This message documents that the data area dump that follows is a data stream that was sent by CA Enterprise Tester to the TSO terminal.

PTE40003 - ISPLINK DISPLAY FAILED, RC=nn

An ISPF ISPLINK DISPLAY request completed with an unsupported return code, 'nn'. If you receive message PTE40003, please note the sequence of events preceding the message and contact CA Enterprise Tester technical support for assistance.

PTE40004 - TPUT FAILED, RC=nn

An TSO TPUT request completed with an unsupported return code, 'nn'. If you receive message PTE40004, please note the sequence of events preceding the message and contact CA Enterprise Tester technical support for assistance.

PTE40101 - The PF key/command entered is not a valid ProTerm command

An unknown command was entered via a CA Enterprise Tester Interactive System Panel. The KEYS command may be used to display the Keys Panel. The Keys Panel lists all valid CA Enterprise Tester commands.

PTE40102 - A command parameter was not valid for the command entered

A parameter was entered following a CA Enterprise Tester Interactive System primary command. The command either does not support parameters, or the parameter entered is not supported by the command.

PTE40103 - The PLU attempted to send a message, use INVITE command to permit

While sending the prior message to the online system, an attempt by the online system to send an unsolicited message was detected by the CA Enterprise Tester Interactive System. The online system message was rejected because the terminal message was already inbound to the online system. The INVITE command may be used to grant permission to the online system to send the intended message.

PTE40104 - The command was not processed because session not active on Port

A command was entered that required that a session be active on the specified port. This message informs the user that the command was ignored because the required session was not active.

PTE40105 - The Reply to a script WTOR has been processed

A CA Enterprise Tester Interactive System Reply command was accepted and passed on to a CA Enterprise Tester Script.

PTE40107 - Function not available; ProTerm Host Commands are not installed

The function requested can not be performed because the CA Enterprise Tester Host Command Environment is not currently installed at your installation. Refer to the *CA Enterprise Tester System Guide* for installation requirements.

PTE40109 - ERASE command requires that cursor be in an unprotected field

The CA Enterprise Tester Interactive System ERASE command was entered while the cursor was positioned outside of the CA Enterprise Tester session screen display area or while the cursor was positioned on a screen attribute or protected field. Either enter the ERASE command in the primary command field, position the cursor in the field to be erased and press ENTER, or position the cursor in the field to be erased and press a PF Key that has been assigned to the ERASE command.

PTE40201 - Token screen size exceeds terminal size; use a larger display

The screen rows and/or columns of the specified token screen are larger than screen rows and/or columns supported by your terminal display. The token screen may be viewed or updated only using a terminal display whose screen rows and/or columns allow proper display of the token screen.

PTE40202 - The command entered is valid only while a Recording is active

An entered CA Enterprise Tester Interactive System command was ignored because a Recording was not active on the specified Port(s).

PTE40203 - An active recording was ended in response to an END RECORD command

This information only message is issued to confirm the successful execution of the entered CA Enterprise Tester Interactive System END RECORD command.

PTE40204 - A RECORD command was issued; RECORD is already active on the Port

The CA Enterprise Tester Interactive System has ignored a RECORD command because a recording is already active on the specified port.

PTE40205 - A recording was started in response to a RECORD command

This message is issued by the CA Enterprise Tester Interactive System simply to inform the user that the a "Record" command has been accepted and that a recording has been successfully started.

PTE40206 - Data set must be pre-allocated and cataloged

CA Enterprise Tester Interactive System data sets used to trace session screens or to record session scripts must be existing cataloged data sets. A data set may be created via the ISPF UTILITIES "DATA SET UTILITY" Panel. The CA Enterprise Tester Interactive System Record and Trace Panels specify required data set characteristics.

PTE40207 - Data set allocation failed with error code xxxxxxxx

The allocation of a data set specified via the CA Enterprise Tester Interactive System Record or Trace Panel was unsuccessful. In the message, "xxxxxxxx", is replaced by the dynamic allocation return code. Dynamic allocation return codes (DAIR) may be found in "Appendix A" of the ISPF Tutorial, which may be accessed by pressing the HELP PF Key while viewing the "ISPF/PDF PRIMARY OPTION MENU".

PTE40208 - The command entered is not a valid command at this panel

An invalid command was entered or a command was entered that was not valid at the panel displayed. The command was not processed.

PTE40301 - The dataset must be a sequential or partitioned dataset

The data set specified via the CA Enterprise Tester Interactive System Play, Analyze, Record or Trace Panel is not a sequential or partitioned data set. A played script must reside in a sequential data set or must be a member in a partitioned data set. The recorded script or a session screen trace must be written to a sequential data set or to a member of a partitioned data set. Specify an acceptable data set.

PTE40302 - The dataset must be fixed length record format (RECFM=F/FB/FBA)

The record format (RECFM) of the data set specified via the CA Enterprise Tester Interactive System Play, Analyze, Record or Trace Panel is not acceptable for the purpose specified. The record format of a data set in which a script is recorded or from which a script is played must be fixed, blocked, RECFM=FB. The record format of a data set to which session screens are traced must be fixed, RECFM=F (may also be FA or FBA). Specify a data set that has an acceptable RECFM.

PTE40303 - The dataset record length (LRECL) must be 'len'

The record length (LRECL) of the data set specified via the CA Enterprise Tester Interactive System Play, Analyze, Record or Trace Panel is not acceptable for the purpose specified. When the message is displayed, 'len' is replaced by the preferred LRECL. The record length of a data set in which a script is recorded or from which a script is played must be 80 (LRECL=80). The record length of a data set to which session screens are traced must be at least 133, but must not exceed 256 (the preferred LRECL is 135, which allows for an ANSI carriage control character, 2 characters for screen borders (sides) and 132 characters for the largest 3270 screen width.) Specify a data set that has an acceptable record length. The record length for a CA Enterprise Tester Log dataset must be 151.

PTE40306 - A PDS member name is acceptable only if data set is partitioned

The organization of the data set specified via the CA Enterprise Tester Interactive System Play, Record, Trace, Analyze or Log Panel is sequential, but a PDS member name was specified with the request. Either specify a partitioned data set or repeat the request after erasing the specified member name.

PTE40308 - A Played Script or session analyzer was ended by a command or verb

This information only message confirms that a played script or session analyzer was ended by the execution of a CA Enterprise Tester Interactive System END command or was ended by the execution of a Script Language END statement.

PTE40309 - A command could not be processed because a script is not active

A CA Enterprise Tester Interactive System command was entered that is active only while a script is active on the port. For example, an EXECUTE command can not be executed unless a script is active on the port. The information only message informs the user that the command was not executed.

PTE40401 - The auto-INVITE time interval has been updated

An Interactive System INVITE command was entered that specified a time interval in the form HHMMSSTH. The interval has been remembered and will be used during subsequent interactive use of CA Enterprise Tester Sessions to anticipate outbound messages that are sent by an online system after the online system has sent a message that specified that the terminal keyboard be freed.

PTE40403 - No session screens have been saved for session Review

The CA Enterprise Tester Interactive System REVIEW command was entered on a port for which no session screen images have been saved for review. The information only message informs the user that the command was not executed.

PTE40405 - ProTerm script execution initiated by Play or Analyze command

A CA Enterprise Tester script was executed using Interactive System Analyze or Play Panels. This information only message informs the user that script execution has been successfully initiated.

PTE40406 - An attempt to ATTACH a session subtask failed

An attempt by the CA Enterprise Tester Interactive System to create a subtask failed. This error should never happen. If problem is not understood or persists, refer all available documentation including abend messages to a CA Enterprise Tester product support representative.

PTE40407 - STOP stopped the specified function(s) on the specified port(s)

One or more scripts, recordings and/or traces, as specified via STOP command specifications, has been stopped. This information only message is issued to conform the success of the STOP request.

PTE40408 - RESUME resumed the specified function(s) on the specified port(s)

One or more scripts, recordings and/or traces, as specified via RESUME command specifications, have been resumed. This information only message is issued to confirm the success of the RESUME request.

PTE40409 - The current review screen is the oldest screen saved for review

A CA Enterprise Tester Interactive System REVIEW command was entered while reviewing the oldest screen that has been saved for review. The REVIEW command becomes inoperable when the oldest REVIEW screen for a port has been reached.

PTE40501 - A session trace was started in response to a TRACE command

A session trace Trace has been started as specified via the Trace Panel. This information only message is issued to confirm that the trace was successfully started.

PTE40502 - To start a session, enter online system name, then press enter

The enter key was pressed while viewing the CA Enterprise Tester System Menu, a session was not started because an online system name or ID was not entered. Enter the online system name in the system ID field provided on the panel, and press enter to start the session.

PTE40503 - A session was requested on a port that has an active session

A SESSION command was entered via the CA Enterprise Tester Interactive System Control Panel that specified both an online system name and a CA Enterprise Tester Port. The specified session could not be started because a session is already active on the specified port. Specify a different port for the new session or specify the port only to select the existing session.

PTE40504 - Specify Port number or identify applicable Port using cursor

A SESSION, PLAY, ANALYZE, MONITOR, RECORD or TRACE command was entered via the Interactive System Control Panel. For a SESSION or MONITOR command, a port must be specified to identify which active Session or script is to be selected. A PLAY, ANALYZE, RECORD or TRACE command must specify the port on which the feature is to be initiated.

PTE40506 - A TRACE command was issued, but TRACE is already active on the Port

The CA Enterprise Tester Interactive System has ignored a TRACE command because a Trace is already active on the specified port.

PTE40507 - A session screen trace was terminated per entered END TRACE command

This information only message is issued to confirm the successful execution of an END TRACE command.

PTE40508 - The command entered is valid only while a TRACE is active

An entered CA Enterprise Tester Interactive System command was ignored because a trace was not active on the specified port(s).

PTE40509 - 'library' allocation failed - xxxxxxxx

The allocate for the session analyzer script library failed. The likely cause is that the CA Enterprise Tester Options Table PTEOM ANALYZE statement specifies the wrong script library. Contact the system programmer responsible for the installation and maintenance of CA Enterprise Tester. In the message, "xxxxxxxx", is replaced by the dynamic allocation return code. Dynamic allocation return codes (DAIR) may be found in "Appendix A" of the ISPF Tutorial, which may be accessed by pressing the HELP PF Key while viewing the "ISPF/PDF PRIMARY OPTION MENU".

PTE40601 - User ID must be 2 thru 8 national characters

A user ID was entered via the CA Enterprise Tester Code Panel. The user ID can not be used to encode data because it consists of other than 2 through 8 national characters. Refer to "Code Security" in topic 4.2 for a full description of the requirements of CA Enterprise Tester Security Code features.

PTE40602 - TSO input input invalid. Tget RC not 0, 8 or 24.

Terminal input received by the CA Enterprise Tester Interactive System from a TSO user's terminal was ignored because the input did not fit the current CA Enterprise Tester session screen. The input contained a 3270 SBA that was invalid or did not correspond to a screen field currently contained in the virtual terminal screen buffer.

PTE40603 - JUMP (=) commands will now terminate sessions and scripts

A "JUMP ON" command was entered by an Interactive System terminal user. When subsequent RETURN or ISPF jump (=) commands are entered, CA Enterprise Tester will automatically terminate all active CA Enterprise Tester functions, including sessions and plays, prior to returning to ISPF.

PTE40604 - JUMP (=) commands will no longer end sessions or scripts

A "JUMP OFF" command was entered by an Interactive System terminal user Subsequent attempts to exit CA Enterprise Tester will be rejected if a CA Enterprise Tester session or script is active. While JUMP is OFF, active sessions and scripts must be ended prior to entering END, RETURN or = commands to exit CA Enterprise Tester.

PTE40605 - Enter "e s", "e p", or change option to end automatically

An END, RETURN or ISPF jump (=) command was entered to exit CA Enterprise Tester. The command was ignored because the CA Enterprise Tester JUMP control is "off" and a CA Enterprise Tester session or script is active. Either turn JUMP on by entering the command "JUMP ON", and repeat the exit request, or terminate all active sessions and scripts.

PTE40606 - Input received via Session Panel does not fit virtual terminal

Input received via the CA Enterprise Tester Session Panel does not match the virtual screen format from which the session display was created. If the problem persists, contact CA Enterprise Tester technical support for assistance.

PTE40607 - The cursor is positioned at the indicated row and column

The Interactive System SPOT command was entered. The current cursor position within the CA Enterprise Tester session screen was displayed in the upper right corner of the CA Enterprise Tester Session Panel as "Cursor at <r,c>". The cursor was positioned at row "r" and column "c".

PTE40608 - Review screens will be limited to the entered number

An REVIEW command was entered via the CA Enterprise Tester Control Panel. The command specified the number of review screens to be saved for each CA Enterprise Tester session. Review screens will subsequently be limited to the number of screens specified.

PTE40609 - the display of several session steps is now complete

A STEP command that specified more than one session step, was processed while a step display interval was in effect. The automatic display of the specified number of session steps is now complete.

PTE40703 - Library allocation failed with error code eeeerrrr

The allocation of a CA Enterprise Tester Script Library, specified via the Interactive System Play Panel, or Analyze Panel Mask Library field. was not successful. In the message, "eeeerrrr", is replaced by the dynamic allocation error (eeee) and reason (rrrr) code. Dynamic allocation error codes may be found in Appendix A of the ISPF tutorial that is presented when HELP is invoked from the ISPF/PDF Primary Option Menu.

PTE40704 - A specified LINK Library is not a Partitioned Data Set

A data set, entered as a CA Enterprise Tester Script Library via the Interactive System Play Panel or Analyze Mask Library field, is not a partitioned data set. Only a PDS with DCB=(RECFM=FB,LRECL=80,...), may be used as a CA Enterprise Tester Script Library.

PTE40705 - Library concatenation failed with error code eeeerrrr

An attempt to concatenate the CA Enterprise Tester Script Libraries that were entered via the Interactive System Play Panel, failed. In the message, "eeeerrrr", is replaced by the dynamic concatenation error (eeee) and reason (rrrr) code. Refer to IBM documentation of DYNALLOC (SVC 99), for dynamic concatenation error and reason codes.

PTE40706 - The specified nnnnnnnn data set name is missing or invalid

A data set name specified via a CA Enterprise Tester Interactive System Panel is invalid or a required data set was not specified. In the message, "nnnnnnnn", is replaced by one of the names, RECORD, TRACE, SCRIPT or LIB n, indicating which data set name is missing or invalid. Correct or enter the data set name.

PTE40707 - The value entered for the oooooooo option was invalid

An invalid value was specified for one of the options listed on the CA Enterprise Tester Interactive System Options Panel. In the message, "oooooooo", is replaced by one of the option names ZOOM, JUMP, PREFIX, REVIEW, LOG or INVITE, identifying the option that was in error. The last valid option value is retained. Enter a correct option value if another option is desired.

PTE40708 - A screen trace caption was accepted or was reset

A TRACE command, that specified a literal trace caption, has been accepted. The caption will be included on subsequently traced screen images.

PTE40709 - A specified script statement has been recorded

A script statement, specified via a RECORD command that specified literal statement data, has been written to the active session recording.

PTE40801 - The review of several session steps is now complete

A REVIEW command that specified more than one session step, was processed while a step display interval was in effect. The automatic display of the specified number of session steps is now complete.

PTE40802 - Roll failed because no corresponding session screen exists

A ROLL command was entered while reviewing a trace screen. The command could not be processed because a session review screen that corresponds to the current trace review screen is not available.

PTE40803 - Roll failed because no corresponding trace screen exists

A ROLL command was entered while reviewing a session screen. The command could not be processed because a trace review screen that corresponds to the current session review screen is not available.

PTE40804 - A data set name must be entered to perform the requested function

A data set name, either a target dataset for a session recording, trace or log, or a source data set containing the script to be played or the trace or trace list to be analyzed, must be specified in order to perform the specified function.

PTE40805 - The system ID entered exceeds 8 characters in length

The online system name entered via the CA Enterprise Tester System Menu System ID field exceeds 8 characters. Valid online system names or IDs are 1 through 8 characters long.

PTE40806 - The _____ command is not appropriate at this panel

This message informs the user that the command named in the message was entered, but that the command is not supported while using the panel currently displayed.

PTE40807 - SENSE=ssss, was sent in response to invalid or unsupported data

A message was received on a CA Enterprise Tester session from an online system. The message contained an invalid 3270 command, order or attribute. A negative response, containing SNA sense code "ssss", was sent to the online system. If the problem persists, a log may be started, and the transaction may be repeated to obtain a hexadecimal dump of the rejected session data.

PTE40808 - A command, script or session was stopped or timed out by ATTN key

The Attention or PA1 key was pressed by the terminal user. This message acknowledges that the Attention key signal has been received and processed by CA Enterprise Tester. Any active function on which the user was waiting has now been stopped.

PTE40809 - End Log command ignored because a log is not active

An END LOG command was entered. This message informs the user that a log was not active and therefore could not be ended.

PTE40901 - Logging was previously started and is currently active

The user entered the LOG command, causing the Log panel to be displayed. This message is issued to inform the user that a log has already been started and is currently active. Logging may be resumed by simply pressing the enter key, or logging may be modified by turning the request unit log on or off before pressing enter.

PTE40902 - Logging was successfully started

The enter key was pressed at the Interactive System Log panel. This message informs the user that the parameters entered were valid and that logging has been successfully started.

PTE40903 - Specified member was not found in specified PDS

A partitioned data set member was entered at the CA Enterprise Tester Play or Analyze panels that was not found in the specified partitioned data set or library (PDSE). You may leave the member name blank or specify a member name pattern to display a list from which a member may be selected.

PTE40904 - Logging was successfully ended

The END command was entered at the Log panel while a log was active, or an END LOG command was entered from another panel. This message informs the user that logging has now been successfully ended.

PTE40905 - Change invalid while log is active or change was invalid

A Log panel field, data set name, sysout class or RU log request field was changed while a log was already active. The specified parameter may be changed only when starting a log. This message may also be issued if an invalid change is made to a Log panel field while logging is active. To change the specified parameter, end and restart logging.

PTE40906 - Enter Monitor or Play command when Number of Ports > 1

A multi-port play request was entered, however enter was pressed without entering a command or a command or PF key was pressed that specified a command other than Monitor (M) or Play (P). Multi-port requests may be started only when the Monitor or Play command is also entered.

PTE40907 - Specify log dsn, dsn and member or sysout, one only

The enter key was pressed at the Interactive System Log panel to start a log. Both the data set name field and the SYSOUT class field contained data. Only one type of log data set may be specified. Either erase the data set name field or erase the SYSOUT class field.

PTE40908 - Logging was resumed - any change to ru logging is active

The enter key was pressed at the Log panel. A log was already active when the Log panel was displayed. This message informs the user that logging has been resumed. If RU session data request field was changed while viewing the panel, that change is now in effect.

PTE40909 - Cursor not PEN field

A CA Enterprise Tester PEN command was entered via a command field or PF key while the cursor was positioned at a displayed session field other than a pen detectable field. The PEN command simulates the operation of a light pen or cursor select key and operates only when the cursor is positioned at a pen-detectable screen field.

PTE41001 - The value entered in the RU log option field was invalid

An invalid value was specified in the RU session data option on the Interactive System Log panel. "N" specifies that session RU data is not to be logged. "Y" may be entered to request that an 'RU Log" be captured. An RU Log is a log containing descriptive messages and a hexadecimal dump of the inbound and outbound 3270 data streams and other SNA commands that flow on a CA Enterprise Tester session and are useful in analyzing sessions at a very technical level.

PTE41003 - Trace data set or data sets in trace list are empty

The data set specified in an Interactive System Browse request or the data set(s) specified in a trace list specified in a Browse request contained no screen images.

PTE41004 - The last screen in trace data set or trace list is displayed

While browsing trace data, the end of the trace data was encountered while attempting to get the next or a specified screen image as specified by a STEP, WAIT or LOCATE command. The message indicates that the last screen image in the trace data set or last screen image in the last data set specified in a trace list is currently displayed.

PTE41005 - Another function is active on the specified port

An attempt was made to start a session, play a script, analyze a session, record a session or trace a session on a port on which a browse was active, or an attempt was made to browse trace data using a port on which a session, played script, session analyzer, recording or trace was active. On a single port, the browse function is mutually exclusive with sessions, scripts, recordings and traces. Another port may be used to start the desired function.

PTE41006 - A trace or trace list browse was ended by an END command

This information-only message confirms that an active browse was ended as requested by an end browse command.

PTE41007 - Command specified Browse, but a trace browse is not active

An END command specified that a browse was to be ended. The command could not be processed because a browse was not active on the port being used.

PTE41008 - The first screen in trace data set or trace list is displayed

While browsing trace data, the beginning of the trace data was encountered while attempting to get the previous screen image in response to a REVIEW command. The message indicates that the first screen image in the trace data set or first screen image in the first data set specified in a trace list is currently displayed.

PTE41009 - The cursor must be within session display for SPOT command

A SPOT command was entered to obtain the row and column at which the cursor is positioned relative to a displayed virtual screen image, however when the command was entered, the cursor was not positioned at a location within the currently displayed virtual terminal screen image. When the SPOT command is entered in the command field or when a PF key assigned to the SPOT command is pressed, position the cursor at a desired virtual terminal screen location before pressing the PF key or the enter key.

PTE41101 - A command specified Play but a played script is not active

An END, STOP or RESUME command was entered to end, stop or resume a played script. The command could not be processed because a played script was not active on the port being used.

PTE41102 - A command specified Analyze but a session analyzer is not active

An END, STOP or RESUME command was entered to end, stop or resume a session analyzer. The

PTE41103 - A data set name or SYSOUT=class,DEST=dest must be specified

A data set name or sysout class was not specified in a CA Enterprise Tester Interactive System trace or record request. Enter a target data set name, data set and member name, SYSOUT=class, or SYSOUT=class,DEST=destname.

PTE41104 - When target is SYSOUT, a member name may not be specified

A CA Enterprise Tester Interactive System trace or record request specified SYSOUT=class in the data set name field. However the request also specified a PDS member name. If the trace or record target is SYSOUT, erase the member name field. If the trace or record target is a PDS, specify the data set name.

PTE41106 - Enter a valid select code: S (Select), B (Browse), E (Edit)

A member list line command other than S for Select, B for Browse, or E for Edit was entered. Enter a valid list line command or press the end key to exit the displayed member list.

PTE41107 - The member name specified has invalid member name characters

The member name specified contained other than aplhameric or national characters, @, #, or $, or the first character of the member name was numeric. Enter a valid partitioned dataset member name to proceed.

PTE41108 - Confirm or cancel request as explained on the displayed panel

A multi-play or other request that requires confirmation of the request was entered and a confirmation panel was displayed stating what to do to confirm your request. Follow the displayed instructions to confirm your request.

PTE41109 - The dataset name specified contains invalid DSN characters

The dataset name specified contained other than aplhameric or national characters, @, #, or $, separated into dataset name nodes using periods, or a valid relative generation number contained in parenthesis immediately following the data set name. Specify a valid dataset name to proceed.

PTE41201 - Delay percent must be a number from 0 through 999'

A Delay percent was entered via the CA Enterprise Tester Play panel that was not a number from 0 to 999. Enter a valid delay percent to factor DELAY statement times contained in a played script by the specified percentage. Enter a percent less than 100 to use only a fraction of DELAY statement time and to make the script run faster. Specify a percent greater than 100 to increase Delay statement time and to make the script run slower.

PTE41202 - Think time must be ssth, ss=sec, t=tenths, h=hundredths

A Think time was entered via the CA Enterprise Tester Play panel that was not in the format 'ssth', where ss is seconds, t is tenths of seconds and h is hundredths of a second. A specified think time delay is incurred during execution of a played script just prior to executing each and every TYPE statement executed, and may be specified to control the speed of a script or session. Erase the think time field or specify a valid think time in the form, ssth, to proceed.

PTE41203 - Number of ports must be 1 through the maximum shown

A value was entered in the Number of Ports field of the CA Enterprise Tester Play panel, however the number was not 1 through the number of ports authorized and displayed on the play panel. Erase the field or enter an '1' to play a script on just one port, or enter an allowable number to play a script on a desired number of ports to perform system stress testing or concurrency testing.

PTE41204 - Assign only ProTerm Commands to Window and Zoom PF keys

A command not directly supported by CA Enterprise Tester was entered in a Window or Zoom Command PF key field and was ignored. To assign an ISPF or other command to a PF key that is not overridden by a CA Enterprise Tester command, assign the command as an ISPF command and erase the Window command. CA Enterprise Tester commands are listed on the keys panel.

PTE41205 - Request was not confirmed; the token was not 'action'

A confirmation panel was displayed following a request to assign a value to, change the value of or to delete a CA Enterprise Tester security token. However, the request was not confirmed by following the instructions displayed in the confirmation window, and the original request, 'action', was not performed.

PTE41206 - Reentered token data did not match previously entered data

New token data was entered in a password field of a token screen display to assign a value to or to change the value of an existing CA Enterprise Tester security token. However, the same data, including cursor position, was not reentered to confirm the value to be assigned to the token. The request to assign or change a token value was aborted. You may assign or change the token value by again entering the desired data and by entering the same data when requested to confirm the data entered.

PTE41207 - The entered data was filed as the value of token xxxx

After entering data into the password field of a CA Enterprise Tester token security screen and after confirming the data entered, the data was filed as the new value of the specified token, 'xxxx', and will subsequently be keyed in when a script is played or a session trace is analyzed that refers to the token.

PTE41208 - Enter D in the field provided to delete token xxxx

While viewing a CA Enterprise Tester token security screen display, data was entered in the delete request field displayed in a window on the panel, however, the data entered was not the character 'D'. If you wish to delete a token, enter 'D', otherwise do not enter data in the delete request field.

PTE41209 - Token xxxx deleted

Following the entry of a token delete request via the delete request field of a CA Enterprise Tester token screen display, and following confirmation of the delete request, the token was deleted, including the token name and data associated with the token, and the token screen image, if the screen image was displayed from your token dataset.

PTE50001 - RETURN STATE IS state (flags)

This message documents the state of a CA Enterprise Tester logical unit after processing major session events, including session initiations, terminal inputs, online system responses and session terminations. When the message is issued, "state" is replaced by one of the state names documented in "CA Enterprise Tester Logical Unit States" in topic 14.2, and "flags" is replaced by a three-byte hexadecimal value that provides additional information about a logical unit or session, as documented in "CA Enterprise Tester Session Status Flags" in topic 14.3.

PTE50002 - PTCC CB INVALID, ABENDING

At the entry to CA Enterprise Tester module, PTECS, a major CA Enterprise Tester control block, the PTCC, was found to be invalid. The control block is documented in the *CA Enterprise Tester System Guide* , for use in CA Enterprise Tester macro level assembler applications. If the message is incurred while using another type of CA Enterprise Tester application, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50003 - PTCC FUNCTION INVALID, ABENDING

At the entry to CA Enterprise Tester module, PTECS, the function code contained in the PTCC control block was found to be invalid. The PTCC control block is documented in the *CA Enterprise Tester System Guide* , for use in CA Enterprise Tester macro level assembler applications. If the message is incurred while using another type of CA Enterprise Tester application, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50005 - PROCESSING LOGON REQUEST

This information-only message documents a call to the CA Enterprise Tester Communication System LOGON routine. The LOGON routine is used by CA Enterprise Tester applications to start a session between a CA Enterprise Tester virtual terminal and an online system.

PTE50006 - OPENING ACB FOR SLU slu

This information-only message documents an attempt by CA Enterprise Tester to activate a virtual terminal for use in a CA Enterprise Tester session. When the message is issued, "slu", is replaced by the VTAM APPL name of the virtual terminal specified by or allocated to the CA Enterprise Tester application.

PTE50007 - ISSUING SETLOGON

This information-only message documents the execution of a VTAM SETLOGON macro statement, to inform VTAM that the virtual terminal is prepared to accept sessions.

PTE50008 - SLU (TERMINAL) APPLID REQUIRED

A CA Enterprise Tester macro level assembler program issued a CA Enterprise Tester LOGON request, as documented in the *CA Enterprise Tester System Guide* , however did not specify the name of the VTAM APPL to be used as a virtual terminal.

PTE50009 - PLU (ONLINE SYSTEM) APPLID REQUIRED

A CA Enterprise Tester macro level assembler program issued a CA Enterprise Tester LOGON request, as documented in the *CA Enterprise Tester System Guide* , however did not specify the VTAM APPL name of the online system with which a session was desired.

PTE50010 - GEN ACB ERROR

A VTAM GENCB macro was executed by CA Enterprise Tester to create a VTAM ACB. The macro did not complete successfully. If other messages, displayed or written to the job log or z/OS system log, do not explain the failure, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50011 - GEN NIB ERROR

A VTAM GENCB macro was executed by CA Enterprise Tester to create a VTAM NIB. The macro did not complete successfully. If other messages, displayed or written to the job log or z/OS system log, do not explain the failure, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50012 - GEN RPL ERROR

A VTAM GENCB macro was executed by CA Enterprise Tester to create a VTAM RPL. The macro did not complete successfully. If other messages, displayed or written to the job log or z/OS system log, do not explain the failure, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50013 - SLU ACB OPEN FAILED ACBERFLG xx

CA Enterprise Tester attempted to open a VTAM ACB, to be used as a virtual terminal. The open did not complete successfully. When the message is issued, "xx" is replaced by the one-byte hexadecimal value contained in the ACB field, ACBERFLG. Official documentation of the ACB error flag may be found in an IBM "VTAM Programming" manual. However, when ACBERFLG is 5A, it means that the name of the virtual terminal specified by or allocated to a CA Enterprise Tester application is unknown to VTAM. The problem may be corrected by varying the installed VTAM application major node, that was installed for use by CA Enterprise Tester, active to VTAM, or by coding appropriate TERMINALS statements in the CA Enterprise Tester Options Table as described in the *CA Enterprise Tester System Guide* .

PTE50014 - SETLOGON FAILED, SLU INACTIVE

After successfully opening the VTAM ACB to be used as a virtual terminal, and before issuing a VTAM SETLOGON macro to activate session start up, CA Enterprise Tester detected that the virtual terminal was now "inactive". If other messages displayed or written to the CA Enterprise Tester log, job log or z/OS system log do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50015 - SETLOGON REQUEST REJECTED, RCFB-rcfb SENSE-sense

After successfully opening the VTAM ACB to be used as a virtual terminal, a VTAM SETLOGON macro statement was executed while attempting to start a CA Enterprise Tester session. The request completed with a non-zero return code. Refer to "VTAM Return Codes and Feedback" in topic 14.6 and "SNA Sense Codes" in topic 14.7 for information about the VTAM return and feedback codes and SNA sense codes contained in this message. If the return, feedback and sense codes and other messages displayed or written to the CA Enterprise Tester log, job log or z/OS system log do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50016 - PROCESSING CONVERSE REQUEST

This information-only message documents a call to the CA Enterprise Tester Communication System CONVERSE routine. The CONVERSE routine is used by CA Enterprise Tester applications to send a message from a CA Enterprise Tester virtual terminal to an online system.

PTE50017 - CONVERSE INVALID DURING state STATE

An invalid call was made to the CA Enterprise Tester Communication System CONVERSE routine. The CONVERSE routine is used by CA Enterprise Tester applications to send a message from a CA Enterprise Tester virtual terminal to an online system. The call was invalid because the CA Enterprise Tester virtual terminal keyboard was inhibited or otherwise unavailable to be used to send a message to the online system. When the message is issued, "state" is replaced by one of the state names documented in "CA Enterprise Tester Logical Unit States" in topic 14.2.

PTE50018 - AID KEY IS NOT ENTER, PA, PF OR CLEAR

An invalid call was made to the CA Enterprise Tester Communication System CONVERSE routine. The CONVERSE routine is used by CA Enterprise Tester applications to send a message from a CA Enterprise Tester virtual terminal to an online system. The aid key parameter specified on the call was not a code that represents a keyboard ENTER, PA, PF, or CLEAR key.

PTE50019 - CURSOR POSITION IS NOT WITHIN SCREEN

An invalid call was made to the CA Enterprise Tester Communication System CONVERSE routine. The CONVERSE routine is used by CA Enterprise Tester applications to send a message from a CA Enterprise Tester virtual terminal to an online system. The cursor address specified on the call was larger than current virtual terminal screen size.

PTE50020 - PROCESSING ATTENTION REQUEST

This information-only message documents a call to the CA Enterprise Tester Communication System ATTN routine. The ATTN routine is used by CA Enterprise Tester applications to send an attention key signal from a CA Enterprise Tester virtual terminal to an online system.

PTE50021 - PREVIOUS SIGNAL COMPLETED, RCFB-rcfb SENSE-sense

This information-only message documents the completion of a previously sent attention key signal. If the message contains non-zero VTAM return and feedback codes or SNA sense codes, information about the codes may be found in "VTAM Return Codes and Feedback" in topic 14.6 and "SNA Sense Codes" in topic 14.7.

PTE50022 - SENT SIGNAL REQUEST

This information-only message is issued when an attention key signal has been successfully sent from a CA Enterprise Tester virtual terminal to an online system as requested by a CA Enterprise Tester application.

PTE50023 - SIGNAL NOT SENT, reason

A CA Enterprise Tester application request to send an attention key signal from a virtual terminal to an online system could not be processed for the reason stated in the message. If the reason is "DATA TRAFFIC IS RESET", then the session is in a state where session traffic is not permitted. If the reason is "AWAITING SIGNAL RESPONSE", then the signal request was not sent because the response to a previously sent signal request has not been received. On a real terminal or a virtual terminal, the attention key actually sends a signal only when a response to a previous attention key signal is not outstanding.

PTE50024 - SIGNAL NOT SENT, KEYBOARD LOCKED DURING state STATE

A CA Enterprise Tester application request to send an attention key signal from a virtual terminal to an online system could not be processed because the virtual terminal keyboard is locked. While the keyboard is locked, all keyboard keys, including the attention key are disabled. When the message is issued, "state" is replaced by one of the state names documented in "CA Enterprise Tester Logical Unit States" in topic 14.2.

PTE50025 - SIGNAL REQUEST REJECTED, RCFB-rcfb SENSE-sense

A CA Enterprise Tester application requested that an attention key signal be sent from a virtual terminal to an online system. The VTAM SEND request completed with a non-zero return code. Refer to "VTAM Return Codes and Feedback" in topic 14.6 and "SNA Sense Codes" in topic 14.7 for information about the VTAM return and feedback codes and SNA sense codes contained in this message. If the return, feedback and sense codes and other messages displayed or written to the CA Enterprise Tester log, job log or z/OS system log do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50026 - PROCESSING INVITE REQUEST

This information-only message documents a call to the CA Enterprise Tester Communication System INVITE routine. The INVITE routine is used by CA Enterprise Tester applications to process messages or commands that are sent by an online system to a CA Enterprise Tester virtual terminal.

PTE50027 - PROCESSING LOGOFF REQUEST

This information-only message documents a call to the CA Enterprise Tester Communication System LOGOFF routine. The LOGOFF routine is used by CA Enterprise Tester applications to terminate a session between a CA Enterprise Tester virtual terminal and an online system or to clean up session resources following the loss of a CA Enterprise Tester session.

PTE50028 - WAIT STATE IS state (flags)

This message documents the state of a CA Enterprise Tester logical unit after processing any session related request, event, or message. When the message is issued, "state" is replaced by one of the state names documented in "CA Enterprise Tester Logical Unit States" in topic 14.2, and "flags" is replaced by a three-byte hexadecimal value that provides additional information about a logical unit or session, as documented in "CA Enterprise Tester Session Status Flags" in topic 14.3.

PTE50029 - function RPL COMPLETE RCFB-0000 SENSE-00000000

This information-only message documents the successful completion of a previously issued VTAM request. When the message is issued, "function" is replaced by a keyword that identifies the completed function. Function may be ATTN, +RSP, -RSP or TERM. ATTN means that a response to a previously sent attention key signal has been received. +RSP or -RSP mean that a previously scheduled positive or negative response has now been sent. TERM means that a previously issued TERMSESS request is now complete.

PTE50030 - function RPL ERROR RCFB-rcfb SENSE-sense

This message documents the abnormal completion of a previously issued asynchronous VTAM request. When the message is issued, "function" is replaced by a keyword that identifies the completed function. Function may be ATTN, +RSP, -RSP or TERM. ATTN means that a response to a previously sent attention key signal was not successfully received. +RSP or -RSP mean that a previously scheduled positive or negative response was not successfully sent. TERM means that a previously issued TERMSESS request failed. Refer to "VTAM Return Codes and Feedback" in topic 14.6 and "SNA Sense Codes" in topic 14.7 for information about the VTAM return and feedback codes and SNA sense codes contained in this message. If the return, feedback and sense codes and other messages displayed or written to the CA Enterprise Tester log, job log or z/OS system log do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50031 - state RPL COMPLETE RCFB-0000 SENSE-00000000

This information-only message documents the successful completion of a previous RPL based VTAM request. When the message is issued, "state" is replaced by one of the state names documented in "CA Enterprise Tester Logical Unit States" in topic 14.2, and identifies the function of the completed RPL based VTAM request.

PTE50032 - ATTN INTERRUPT PROCESSED, ACTION IS action

When a CA Enterprise Tester Interactive System user presses the attention key while CA Enterprise Tester is waiting for the completion of a session event, this message documents the interruption of normal session processing and the subsequent action taken by CA Enterprise Tester. When the message is issued, "action" is replaced by one of the keywords WAIT, ATTN, TIMEOUT or LOGOFF. The keywords WAIT, ATTN and TIMEOUT document the command entered by an Interactive System user at the CA Enterprise Tester Attention panel that is displayed when some session events are interrupted. WAIT indicates that the user entered the INVITE command to continue waiting for the completion of the interrupted session event. ATTN indicates that the user entered the ATTN command to send an attention key signal through to the online system. TIMEOUT indicates that the user entered the END command to stop waiting for the completion of the interrupted session event and to return a session timeout condition to the CA Enterprise Tester application. The keyword LOGOFF documents the action taken automatically by CA Enterprise Tester when session initiation or termination is interrupted.

PTE50033 - state RPL ERROR RCFB-rcfb SENSE-sense

This message documents the abnormal completion of a previous RPL based VTAM request. When the message is issued, "state" is replaced by one of the state names documented in "CA Enterprise Tester Logical Unit States" in topic 14.2, and identifies the function of the completed RPL based VTAM request. VTAM return code and feedback values and SNA sense codes contained in the message indicate the reason for the request failure. Refer to "VTAM Return Codes and Feedback" in topic 14.6 and "SNA Sense Codes" in topic 14.7 for information about the VTAM return and feedback codes and SNA sense codes. Some combinations of return codes and sense codes may indicate incomplete or incorrect product installation.

A REQSESS error with RCFB-1012 and SENSE-08570002 means that the specified online system is not currently active.

A REQSESS error with RCFB-1012 and SENSE-087D0001 (RCFB=1453 in some releases of VTAM), means that the online system name specified in a session request is unknown to VTAM, and may be the result of misspelling the name via a SESSION command or LOGON statement. If the name specified can be used to successfully start a native terminal session, it is likely that the name is a short system name that is translated to an actual VTAM APPL name using a VTAM USS table or other network product. If this is the case, SYSTEM statements may be coded in your CA Enterprise Tester Options Table, as documented in the CA Enterprise Tester System Guide, to allow the alias online system name to be used to start a CA Enterprise Tester session.

A REQSESS error with RCFB-144B means that the VTAM logon mode table entry name specified with a session request was not found in the mode table that is associated with the APPL statement that defines the CA Enterprise Tester virtual terminal to VTAM.. The error may result from specifying an invalid name via the SESSION command, Terminal Option panel or LOGON statement terminal model parameter. If a model name is not specified, the problem may be that CA Enterprise Tester virtual terminal APPL statements are associated with the wrong VTAM mode table, or that the mode table entry names specified via the LOGMODES and/or EXTMODES statements contained in your CA Enterprise Tester Options Table are incorrect. Refer to installation instructions and Options Table documentation contained in the *CA Enterprise Tester System Guide* for information about VTAM APPL statements, selecting a VTAM mode table and coding the Options Table LOGMODES and EXTMODES statements.

PTE50034 - REQUESTING SESSION WITH PLU plu, LOGMODE logmode

This information-only message documents the name of an online system, "plu", with which a session is being started and the logon mode table entry name, "logmode", that was specified with the session request.

PTE50035 - SESSION NOT REQUESTED, SLU INACTIVE

After successfully opening the VTAM ACB to be used as a virtual terminal, and before issuing a VTAM REQSESS macro to request a session, CA Enterprise Tester detected that the virtual terminal was now "inactive". The session request was not issued. If other messages displayed or written to the CA Enterprise Tester log, job log or z/OS system log do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50036 - SESSION REQUEST REJECTED, RCFB-rcfb SENSE-sense

While processing a CA Enterprise Tester application LOGON request, and after successfully activating a CA Enterprise Tester virtual terminal, a VTAM REQSESS macro was executed to start a session with an online system. The request completed with a non-zero return code. Refer to "VTAM Return Codes and Feedback" in topic 14.6 and "SNA Sense Codes" in topic 14.7 for information about the VTAM return and feedback codes and SNA sense codes contained in this message. If the return, feedback and sense codes and other messages displayed or written to the CA Enterprise Tester log, job log or z/OS system log do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50037 - PROCESSING BIND CID-cid

This information-only message identifies the SNA BIND request that is being processed. A BIND request contains session parameters that were sent by an online system to a terminal during session start up. When the message is issued, "cid" is replaced by the hexadecimal value of a unique "communication ID", that is assigned by VTAM and that is used to distinguish this session from other sessions in the network. When CA Enterprise Tester request unit logging is active, CID may be used to correlate session messages when a "session" actually consists of multiple sessions that result when an online system passes a session to another system or queues a session request to a virtual terminal that is already in session.

PTE50038 - ACCEPTING BIND'

This information-only message indicates that session parameters contained in a BIND request are acceptable to CA Enterprise Tester and that a VTAM OPNSEC macro statement is about to be executed to complete the start up of a session.

PTE50039 - BIND REJECTED (reason) CID-cid

This message is issued when a BIND is processed that contains unacceptable session parameters. Subsequently, a negative response is sent to the online system to reject the BIND. When an online system issues an OPNDST to send a BIND, it may use session parameters contained in the mode table entry that is used in a session request, or may specify session parameters contained in a terminal definition maintained by the online system. Therefore, the mode table used for CA Enterprise Tester virtual terminals and any definitions used by an online system are likely places to look for the source of the unacceptable session parameters.

When the message is logged, "reason" identifies the unacceptable BIND parameter. BINDs are rejected for the following reasons:

- 'FM PROF NOT 03' means that the BIND contains a function management profile other than X'03'.
- 'TS PROF NOT 03' means that the BIND contains a transmission services profile other than X'03'.
- 'PRI PROT NOT B1' means that the BIND primary LU protocol field contains a value other than X'B1'.
- 'SEC PROT MISSING 90' means that in the BIND secondary LU protocol field, bit 0 or bit 3 (X'90') is not on.
- 'COM PROT NOT 3080' means that the BIND common protocol field contains a value other than X'3080'.

When CA Enterprise Tester request unit logging is active, the CID contained in the message may be used to locate a dump of the BIND that is being rejected.

PTE50040 - LAST QUEUE ITEM IS NOT A BIND

CA Enterprise Tester maintains an internal "bind queue" to deal with passed and queued sessions serially. After processing an UNBIND that specified that a BIND was forthcoming, a request unit other than a BIND was found at the end of the queue. The implication is that consecutive UNBINDs were invalidly received. This message indicates a possible CA Enterprise Tester logic error. Contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE50041 - OPNSEC NOT ISSUED, SLU INACTIVE

Prior to executing a VTAM OPNSEC macro statement to accept a session with an online system, CA Enterprise Tester detected that the virtual terminal was now "inactive". The OPNSEC request was not issued. If other messages displayed or written to the CA Enterprise Tester log, job log or z/OS system log do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50042 - OPNSEC REQUEST REJECTED, RCFB-rcfb SENSE-sense

The CA Enterprise Tester Communication System executed a VTAM OPNSEC macro statement to accept a session between a virtual terminal and an online system. The request completed with a non-zero return code. Refer to "VTAM Return Codes and Feedback" in topic 14.6 and "SNA Sense Codes" in topic 14.7 for information about the VTAM return and feedback codes and SNA sense codes contained in this message. If the return, feedback and sense codes and other messages displayed or written to the CA Enterprise Tester log, job log or z/OS system log do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50043 - SESSIONC NOT ISSUED, SLU INACTIVE

to executing a VTAM SESSIONC macro statement to reject a session with an online system, CA Enterprise Tester detected that the virtual terminal was now "inactive". The SESSIONC request was not issued. If other messages displayed or written to the CA Enterprise Tester log, job log or z/OS system log do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50044 - SESSIONC REQUEST REJECTED, RCFB-rcfb SENSE-sense

After receiving an unacceptable SNA BIND request, as documented in a previously logged message, CA Enterprise Tester executed a VTAM SESSIONC macro statement to send a negative response to the online system. The request completed with a non-zero return code. Refer to "VTAM Return Codes and Feedback" in topic 14.6 and "SNA Sense Codes" in topic 14.7 for information about the VTAM return and feedback codes and SNA sense codes contained in this message.

PTE50045 - SENDING DATA REQ RH-rh

This information-only message documents the bracket, chain and direction flags contained in the Request Header (RH) of a message that is sent from a CA Enterprise Tester virtual terminal to an online system. "SNA Request Header (RH) Codes" in topic 14.4 explains the meaning of the RH string contained in this message. A display of the data contained in the message is logged immediately following this message.

PTE50046 - DATA NOT SENT, DATA TRAFFIC IS RESET

CA Enterprise Tester was unable to send a message from a CA Enterprise Tester virtual terminal to an online system because the session was in a state where session traffic is not permitted. If the problem is reproducible, CA Enterprise Tester session request unit logging may be used to obtain a detailed session trace that will explain the reason for the data traffic reset condition.

PTE50047 - SEND REQUEST REJECTED, RCFB-rcfb SENSE-sense

The CA Enterprise Tester Communication System executed a VTAM SEND macro statement to send a message from a virtual terminal to an online system. The request completed with a non-zero return code. Refer to "VTAM Return Codes and Feedback" in topic 14.6 and "SNA Sense Codes" in topic 14.7 for information about the VTAM return and feedback codes and SNA sense codes contained in this message. If the return, feedback and sense codes and other messages displayed or written to the CA Enterprise Tester log, job log or z/OS system log do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50048 - SENDING SHUT COMPLETE

This information-only message is issued when CA Enterprise Tester sends an SNA SHUTC command from a virtual terminal to an online system. A SHUTC command is sent in response to a previously received SNA SHUTD (shutdown) command, and informs the online system that the requested shutdown is now complete.

PTE50049 - SHUTC NOT SENT, DATA TRAFFIC IS RESET

CA Enterprise Tester was unable to send an SNA SHUTC request from a CA Enterprise Tester virtual terminal to an online system because the session was in a state where session traffic is not permitted. If the problem is reproducible, CA Enterprise Tester session request unit logging may be used to obtain a detailed session trace that will explain the reason for the data traffic reset condition.

PTE50050 - SHUTC REQUEST REJECTED, RCFB-rcfb SENSE-sense

After receiving an SNA SHUTD (shutdown) request, the CA Enterprise Tester Communication System executed a VTAM SEND macro statement to send a SHUTC (shutdown complete) response to the online system. The request completed with a non-zero return code. Refer to "VTAM Return Codes and Feedback" in topic 14.6 and "SNA Sense Codes" in topic 14.7 for information about the VTAM return and feedback codes and SNA sense codes contained in this message. If the return, feedback and sense codes and other messages displayed or written to the CA Enterprise Tester log, job log or z/OS system log do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50051 - BIND QUEUED

When an SNA BIND request is received at a CA Enterprise Tester virtual terminal, the BIND is added to a bind queue. This information-only message documents that a bind has been queued. By queuing binds, CA Enterprise Tester is able to defer bind processing when a bind is received while a session is still active. A queued bind is processed when no session exists or after an active session has ended.

PTE50052 - SDT PROCESSED, RECEIVE SCHEDULED

When a session is initially started, or after processing an SNA CLEAR request, a session enters the data traffic reset state. While a session is in the reset state, normal session messages may not be sent. This information-only message is issued when an SNA Start Data Traffic (SDT) request that is sent by an online system or by VTAM, is processed while a session is in the reset state. When an SDT is processed, CA Enterprise Tester schedules a VTAM RPL exit to receive subsequent normal flow session messages.

PTE50054 - SDT PROCESSED BUT RECEIVE REJECTED, RCFB-rcfb SENSE-sense

When a session is initially started, or after processing an SNA CLEAR request, a session enters the data traffic reset state. While a session is in the reset state, normal session messages may not be sent. After receiving an SNA Start Data Traffic request, CA Enterprise Tester executed a VTAM RECEIVE macro statement, to schedule a receive exit to receive subsequent normal flow session messages. The request completed with a non-zero return code. Because CA Enterprise Tester would be unable to successfully communicate on a session if it is unable to receive normal session messages, the active session is subsequently terminated. Refer to "VTAM Return Codes and Feedback" in topic 14.6 and "SNA Sense Codes" in topic 14.7 for information about the VTAM return and feedback codes and SNA sense codes contained in this message. If the return, feedback and sense codes and other messages displayed or written to the CA Enterprise Tester log, job log or z/OS system log do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50060 - RESPONDING TO READ BUFFER COMMAND

This information-only message is issued after receiving a 3270 read buffer command and before sending the contents of the virtual terminal buffer(s) to the online system.

PTE50061 - RESPONDING TO READ MODIFIED COMMAND

This information-only message is issued after receiving a 3270 read modified or read modified all command and before sending the requested 3270 read modified data stream from the terminal buffer(s) to the online system.

PTE50062 - REPLYING TO READ PARTITION QUERY

This information-only message is issued after receiving a 3270 write structured field command that contained a read partition query structured field, and before sending the requested query response from the virtual terminal to the online system.

PTE50063 - ru-name RU DISCARDED, CID NOT MATCHED

A request received from an online system was discarded by CA Enterprise Tester because session id (CID) accompanying the request did not match the active session id. When PTE50063 is issued, ru-name is replaced by the SNA name of the RU whose CID did not match the current session id.

PTE50064 - ru-name RU DISCARDED AT state STATE

A message that was received from an online system was discarded by CA Enterprise Tester because the message could not be processed correctly given the current state of the CA Enterprise Tester virtual terminal. When the message is issued, "ru-name" is replaced by the SNA name of the discarded RU, and "state" is replaced by one of the state names documented in "CA Enterprise Tester Logical Unit States" in topic 14.2. If other messages, displayed or written to the job log or z/OS system log, do not explain the apparent session failure, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50066 - AWAITING PLU ACTION AFTER -RSP TO DATA

An online system sent a negative response to a message previously sent by CA Enterprise Tester. If not already in a suitable state to receive an expected recovery message from the connected online system (PLU), the CA Enterprise Tester virtual terminal entered a suitable receive state, and waited for the PLU recovery message.

PTE50069 - SIGNAL RESPONSE ACKNOWLEDGED

A response to an SNA SIGNAL request was received at a CA Enterprise Tester virtual terminal. No action is required of or taken by CA Enterprise Tester.

PTE50087 - UNBIND RECEIVED - SESSION ENDED

An SNA UNBIND request was received at a CA Enterprise Tester virtual terminal. An UNBIND request is sent to a terminal when an online system terminates a session, possibly as a result of entering a transaction that requests that a session be ended.

PTE50088 - UNBIND RECEIVED - SESSION PASSED

An SNA UNBIND request, that indicated that a BIND for another session was forthcoming, was received at a CA Enterprise Tester virtual terminal. An UNBIND with bind forthcoming is sent to a terminal when an online system terminates a session by requesting that the session be passed to some other system. When the BIND is received, the virtual terminal will be placed in session with the online system that sends the BIND.

PTE50089 - UNBIND CID NOT MATCHED - HALTING

An SNA UNBIND request was received at a CA Enterprise Tester virtual terminal. The UNBIND is invalid because the session id (CID) of the UNBIND RU did not match the id of the currently active session. This condition should never occur. CA Enterprise Tester responds by terminating any session that may be active, and deactivates the virtual terminal. While virtual terminal activity is "halting", session requests will not be accepted.

PTE50090 - CLEANUP RECEIVED - SESSION ENDED

An SNA CLEANUP request was received at a CA Enterprise Tester virtual terminal. A CLEANUP request may be sent to a terminal by VTAM to end a session between the terminal and an online system. A CLEANUP is a possible result of entering a transaction that requests that a session be ended or may be sent if the link that connects a terminal and an online system is lost.

PTE50091 - CLEANUP CID NOT MATCHED - HALTING

An SNA CLEANUP request was received at a CA Enterprise Tester virtual terminal. The CLEANUP is invalid because the session id (CID) of the CLEANUP request did not match the id of the currently active session. A CLEANUP request may be sent to a terminal by VTAM to end a session between the terminal and an online system. A CLEANUP is a possible result of entering a transaction that requests that a session be ended or may be sent if the link that connects a terminal and an online system is lost. This condition should never occur. CA Enterprise Tester responds by terminating any session that may be active, and deactivates the virtual terminal. While virtual terminal activity is "halting", session requests will not be accepted.

PTE50092 - UNKNOWN NOTIFY RECEIVED, HALTING

A network services NOTIFY request was received at a CA Enterprise Tester virtual terminal. However, the notify received was not understood by CA Enterprise Tester to by a notification that a session had ended. CA Enterprise Tester processed the unverified notification of session termination as though it were an understood NOTIFY request. If this message is issued and you do not understand the reason for session termination, please contact CA Enterprise Tester technical support for assistance.

PTE50093 - UNKNOWN RU RECEIVED, RPLCNTRL-xxxxxx RU-data

A unknown request unit was received at a CA Enterprise Tester virtual terminal. Because CA Enterprise Tester does not know how to respond to the request, the session is terminated and the virtual terminal is deactivated. The message contains the three-byte request unit identification contained in VTAM RPL field, RPLCNTRL, and a hexadecimal display of all or the leading part of the request unit data, if a session request unit log is not active. Please contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE50094 - SENDING UNCOND TERMSESS CID-cid

A VTAM TERMSESS macro statement was executed to terminate a session between a CA Enterprise Tester virtual terminal and an online system. The session is being terminated at the request of a CA Enterprise Tester application or following an irresolvable error. The message contains the session id (CID) that is being terminated.

PTE50095 - TERMSESS REQUEST REJECTED, RCFB-rcfb SENSE-sense

The CA Enterprise Tester Communication System executed a VTAM TERMSESS macro statement to terminate a session between a virtual terminal and an online system. The request completed with a non-zero return code. Refer to "VTAM Return Codes and Feedback" in topic 14.6 and "SNA Sense Codes" in topic 14.7 for information about the VTAM return and feedback codes and SNA sense codes contained in this message. If the return, feedback and sense codes and other messages displayed or written to the CA Enterprise Tester log, job log or z/OS system log do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50096 - CLOSING SLU ACB

This information-only message is issued when a CA Enterprise Tester virtual terminal is deactivated by closing the ACB that is used to access VTAM services.

PTE50099 - SENT +RSP TO ru-name

A message sent by an online system was received at a CA Enterprise Tester virtual terminal and was successfully processed. The received message requested a positive response upon successful processing of the message. When PTE50099 is issued, ru-name, is replaced by the SNA name of the outbound message that requested a positive response. This information-only message indicates that the requested positive response was sent to the online system.

PTE50100 - ru-name +RSP NOT SENT, DATA TRAFFIC IS RESET

After successfully receiving and processing a message that was sent from an online system to a virtual terminal, CA Enterprise Tester was unable to send a requested positive response because the session had ended or data traffic had been reset. When message PTE50100 is issued, ru-name is replaced by the SNA name of the RU to which a positive response could not be sent. If a log was active, additional messages document events that caused the data traffic reset condition.

PTE50101 - ru-name +RSP SEND FAILED, RCFB-rcfb SENSE-sense

After successfully receiving and processing a message that was sent from an online system to a virtual terminal, CA Enterprise Tester attempted to send a requested positive response to the online system. When message PTE50101 is issued, ru-name is replaced by the SNA name of the RU to which a positive response was being sent. The VTAM SEND request completed with a non-zero return code. Refer to "VTAM Return Codes and Feedback" in topic 14.6 and "SNA Sense Codes" in topic 14.7 for information about the VTAM return and feedback codes and SNA sense codes contained in this message. If the return, feedback and sense codes and other messages displayed or written to the CA Enterprise Tester log, job log or z/OS system log do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50102 - SENDING -rsp sense TO ru-name RU - reason

A message that was sent from an online system to a CA Enterprise Tester virtual terminal was unacceptable either because the message was invalid or because a contention situation was won by the virtual terminal. PTE50102 is issued prior to sending a negative response to the online system. When the message is issued, -rsp, is replaced by either '-RSP' or '-LUS' if protocol requires that a negative response be sent using an SNA LUSTAT command, sense is replaced by an SNA sense code (see below) that defines the reason for the negative response, ru-name is replaced by the SNA name of the RU to which the negative response was sent, and reason is replaced by a description of the reason for the negative response.

- SENSE-0813, 'SLU WON CONTENTION', means that a message had been sent to the online system prior to receiving a message from the online system that contained a begin bracket indicator. SNA protocol dictates that when both session partners send a message while a session is in the contention state, that the message sent by the secondary logical unit (a terminal), is to be accepted by the online system and that the message sent by the online system is to be rejected by the terminal.

- SENSE-081B, 'INBOUND OR KB BUSY', means that data had already been keyed into the virtual terminal when a message containing a begin bracket indicator was received from an online system. By rejecting the online system begin bracket request, as prescribed by contention state protocols, terminal user keystrokes are protected until the user presses enter to send the terminal input.
- SENSE-0829, 'READ REQUIRES CD', means that an online system sent a 3270 read buffer, read modified or read modified all command or sent a read partition query structured field, however did not send a change direction indicator allowing the virtual terminal to send the requested reply.
- SENSE-0863, 'CHAR SET ERROR', means that a 3270 Set Attribute (SA), Start Field Extended (SFE), or Modify Field (MF) order was processed that contained a matrix character set attribute that referred to a programmed symbol set that is not supported by CA Enterprise Tester. Normally, CA Enterprise Tester supports only character set attributes that specify character attribute reset and SA orders that refer to character set, X'F1' (the APL/Text symbol set). The CA Enterprise Tester Options Table PTEOM CONTROL statement may be changed to specify CS=YES, to instruct CA Enterprise Tester to tolerate SA,SFE and MF orders that refer to symbol sets. Tolerating symbol sets means that orders that refer to symbol sets will not be rejected.
- The two-byte hexadecimal number that follows sense 0863, is the offset of the 3270 order that was rejected, relative to the request unit that contains the order. When a CA Enterprise Tester log is active, the offset may be used to find the rejected order in the logged display of the rejected request unit. For example, sense 0863801C, indicates that a 3270 order at offset X'001C' in the request unit being processed contains an unsupported reference to a symbol set.
- SENSE-1001, 'RU DATA ERROR', means that an invalid 3270 command or order was found in the rejected request unit. 3270 commands or orders are invalid when an operand of the command or order is invalid or when a command or order at the end of a request unit chain is truncated. For example, an SBA or RA order is invalid if the operand does not contain a valid 3270 buffer address, or if the order is at the end of the RU chain and the required buffer address is truncated.

The two-byte hexadecimal number that follows sense 1001, is the offset of the 3270 command or order that was rejected, relative to the request unit that contained the command or order. When a CA Enterprise Tester log is active, the offset may be used to find the rejected command or order in the logged display of the rejected request unit. For example, sense 1001801C, indicates that a 3270 order at offset X'001C' in the request unit being processed contains an invalid or truncated operand.

- SENSE-2002, 'CHAINING ERROR', means that a data request is being rejected because it contains invalid chain indicators. A chain may contain a single request unit that specifies only-in-chain (OIC). A multiple request unit chain must specify first-in-chain (FIC) on the first request unit in the chain, last-in-chain (LIC) on the last request unit in the chain and must specify middle-in-chain (MIC) on all request units between the first and last request unit in the chain. A chaining error is recognized when any other arrangement of chain indicators is found.
- SENSE-2003, 'BRACKET STATE ERROR' means that a data request was received at a virtual terminal while the session was between brackets, but the request did not specify begin bracket.
- SENSE-2004, 'DIRECTION ERROR' means that a data request was received at a virtual terminal while the session was in bracket with inbound direction.
- SENSE-2005, 'DATA TRAFFIC RESET' means that a data request was received at a virtual terminal while the session was in the data traffic reset or shutdown state.
- SENSE-4003, 'BB BUT NOT FIC/OIC' means that a data request was received at a virtual terminal that specified begin bracket but did not also specify first in chain or only in chain.
- SENSE-4007, '+RSP REQ ON NON-LIC' means that a data request was received at a virtual terminal that requested a positive response, however the request was not the last or only element of the chain.
- SENSE-4009, 'CD FOUND ON NON-LIC' means that a data request was received at a virtual terminal that contained a change direction indicator, however the request was not the last or only element of the chain.

PTE50103 - ru-name -rsp NOT SENT, SESSION NOT ACTIVE

A message that was sent from an online system to a CA Enterprise Tester virtual terminal was unacceptable either because the message was invalid or because a contention situation was won by the virtual terminal. A negative response to the request was not sent to the online system because CA Enterprise Tester detected that the session had been terminated as a result of another asynchronous event. When PTE50103 is issued, ru-name is replaced by the SNA name of the RU to which a negative response was being sent, and -rsp is replaced by -RSP if a negative response was being sent or by '-LUS' if the negative response was being sent via an SNA LUSTATUS command. If other messages displayed or written to the CA Enterprise Tester log, job log or z/OS system log do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50104 - ru-name -rsp SEND FAILED, RCFB-rcfb SENSE-sense

A message that was sent from an online system to a CA Enterprise Tester virtual terminal was unacceptable either because the message was invalid or because a contention situation was won by the virtual terminal. When PTE50104 is issued, ru-name is replaced by the SNA name of the RU to which a negative response was being sent, and -rsp is replaced by '-RSP' if a negative response was being sent or by '-LUS' if the response was being sent via an SNA LUSTATUS command. The attempt to send the negative response to the online system, using a VTAM SEND request completed with a non-zero return code. Refer to "VTAM Return Codes and Feedback" in topic 14.6 and "SNA Sense Codes" in topic 14.7 for information about the VTAM return and feedback codes and SNA sense codes contained in this message. If the return, feedback and sense codes and other messages displayed or written to the CA Enterprise Tester log, job log or z/OS system log do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50107 - RECEIVED exit ru type RH-rh RCFB-rcfb SENSE-sense CID-cid

When session request unit logging is active, this message is logged for each request or response unit (RU) received on a CA Enterprise Tester session. The message provides information about the RU received and precedes a display of request unit data when an RU contains data. When the message is issued, lowercase words in the message text shown above are replaced by values that provide the following information about an RU:

- "exit" is one of the values SC or NS, if CA Enterprise Tester received the RU via Session Control or Network Services exit routines. No value is displayed when a normal flow RU is received.
- "ru" is the name of the RU received. "SNA Request and Response Units (RU)" in topic 14.5 contains a description of each RU. The RU name is "RU-xxxxxx" when an RU is unknown to CA Enterprise Tester, where xxxxxx is the hexadecimal value of the three-byte request unit identification contained in VTAM RPL field, RPLCNTRL.
- "type" is one of the values, REQ or RSP, and indicates whether the RU is a request or a response.
- "rh" is a string of codes that indicate bits that are set in the Request Header (RH) associated with the RU. RH codes are described in SNA Request Header (RH) Codes on page 14-11.
- "rcfb" is the VTAM return code and feedback that was contained in the RPL that was used to receive the RU. Refer to "VTAM Return Codes and Feedback" in topic 14.6for information about VTAM return code and feedback values.
- "sense" is the SNA system sense code and user sense code that was contained in the RPL that was used to receive the RU. Refer to "SNA Sense Codes" in topic 14.7 for information about SNA sense codes.

- "cid" is the "Communication ID" assigned to the session by VTAM and identifies the session on which the RU was received.

PTE50108 - NO MESSAGE TEXT - PTECS+offset

This message is issued when a CA Enterprise Tester routine intended to issue a message, however no message text was provided to the message routine. The message indicates a CA Enterprise Tester logic error. Please contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE50109 - RECEIVE EXIT NOT SCHEDULED, RCFB-rcfb SENSE-sense

After receiving an RU, the CA Enterprise Tester receive exit issued another VTAM RECEIVE macro statement to schedule the exit to receive a subsequent RU. The request completed with a non-zero return code. Because CA Enterprise Tester would be unable to successfully communicate on a session if it is unable to receive normal session messages, the active session is subsequently terminated. Refer to "VTAM Return Codes and Feedback" in topic 14.6 and "SNA Sense Codes" in topic 14.7 for information about the VTAM return and feedback codes and SNA sense codes contained in this message. If the return, feedback and sense codes and other messages displayed or written to the CA Enterprise Tester log, job log or z/OS system log do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in problem resolution.

PTE50110 - SCIP EXIT RU NOT SUPPORTED

Not BIND, SDT, CLEAR or UNBIND The CA Enterprise Tester Session Control exit routine (SCIP) was called by VTAM to process an RU that is unknown to CA Enterprise Tester. Because CA Enterprise Tester does not know how to respond to the request, any active session is subsequently terminated. A display of the unknown request unit is logged when session request unit logging is active. Please contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE50111 - HALT NET ACKNOWLEDGED

The CA Enterprise Tester TPEND exit routine was called by VTAM. The reason code contained in the parameter list was 0 (zero), indicating that a HALT command was entered to terminate VTAM in an orderly fashion. CA Enterprise Tester responds by terminating any active session and by deactivating the CA Enterprise Tester virtual terminal.

PTE50112 - HALT NET,QUICK OR VARY NET,INACT ACKNOWLEDGED.

The CA Enterprise Tester TPEND exit routine was called by VTAM. The reason code contained in the parameter list was 4, indicating that a HALT NET,QUICK command was entered to terminate VTAM or a VARY NET,INACT command was entered for the VTAM APPL being used as a CA Enterprise Tester virtual terminal. CA Enterprise Tester responds by deactivating the CA Enterprise Tester virtual terminal.

PTE50113 - HALT NET,CANCEL OR VTAM ABEND ACKNOWLEDGED

The CA Enterprise Tester TPEND exit routine was called by VTAM. The reason code contained in the parameter list was 8, indicating that a HALT NET,CANCEL was entered to cancel VTAM or that VTAM is terminating abnormally. CA Enterprise Tester responds by deactivating the CA Enterprise Tester virtual terminal.

PTE50114 - SESSION REJECTED, SENSE-sense

A network services NOTIFY request was received at a CA Enterprise Tester virtual terminal. The NOTIFY request indicated a "procedure error" (8106200303). After accepting a CA Enterprise Tester session request, the NOTIFY was sent to the virtual terminal by VTAM when the online system rejected the session request. When the request is received, CA Enterprise Tester cleans up session resources and deactivates the virtual terminal. The message contains the SNA sense code specified by the online system. "SNA Sense Codes" in topic 14.7 contains information about SNA sense codes. While the general meaning of sense codes are defined by SNA, online systems choose the sense code that represents the reason for rejecting a session request. For example, the following are a few sense codes that are used by CICS when a session request is rejected.

- SENSE-0801 is used by CICS when a TCT entry is out-of-service.
- SENSE-080E is used by CICS when a terminal can not be auto installed because the AUTOINSTALL MAX value is zero.
- SENSE-0821 is used by CICS when a terminal can not be auto installed because no TERMINAL and TYPETERM definitions have been defined that match the VTAM logon mode table entry that was used to start a session.

PTE60001 - ACCESS PORT IS NOT * OR GT 0

The first value found following the CA Enterprise Tester ACCESS command was not valid. The operand following an ACCESS command is a CA Enterprise Tester port number, and may be missing to indicate that a port is to be assigned, may be an asterisk, '*', to specify that the port on which the exec is running is to be accessed, or may be an integer that is greater than or equal to 1. A port number may be specified using a variable field whose value is the port number to be accessed. The value specified in the ACCESS command was not a supported value.

PTE60002 - UNKNOWN VALUE FOLLOWS ACCESS PORT

A CA Enterprise Tester ACCESS command was processed that specified a valid port number expression. However, a value was found following the valid port number expression. An ACCESS command may specify no more than one port number expression. The expression may be missing to indicate that a port is to be assigned, may be an asterisk, '*', to specify that the port on which the exec is running is to be accessed, or may be an integer that is greater than or equal to 1. A port number may be specified using a variable field whose value is the port number to be accessed. The value specified following the valid ACCESS port expression is not supported.

PTE60003 - IRXEXCOM DROP FAILED, RC=rc

An attempt to drop a REXX variable field failed. The return code, expressed in hexadecimal, returned by the REXX variable access routine, IRXEXCOM, replaces 'rc' in the message. The IBM publication, 'TSO/E, REXX Reference' (SC28-1883-x) documents IRXEXCOM return codes. However, this error should never occur. If the cause of the problem can not be determined, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE60004 - LOCK PARAMETER NOT SUPPORTED

While processing a CA Enterprise Tester LOCK command, a value was found following the command name, 'LOCK'. The LOCK command supports no parameter values. The LOCK command locks the port that is currently being accessed.

PTE60005 - UNLOCK PARAMETER NOT SUPPORTED

While processing a CA Enterprise Tester UNLOCK command, a value was found following the command name, 'UNLOCK'. The UNLOCK command supports no parameter values. The UNLOCK command releases any lock on the port that is currently being accessed.

PTE60006 - SIGNAL CONDITION IS NOT OFF, ERROR, FAILURE OR HALT

A value was found following the CA Enterprise Tester SIGNAL command that was other than one of the supported keywords, OFF, ERROR, FAILURE or HALT. The keyword OFF indicates that no REXX condition is to be raised for the CA Enterprise Tester condition identified by the SIGNAL command. The keywords ERROR, FAILURE and HALT are REXX conditions that may be detected using REXX SIGNAL and CALL instructions, and that are to be raised by CA Enterprise Tester when an identified CA Enterprise Tester condition occurs. No other values may be coded immediately following the CA Enterprise Tester SIGNAL command name.

PTE60007 - "ON" DOES NOT FOLLOW SIGNAL CONDITION

A CA Enterprise Tester SIGNAL command was processed that did not contain the required keyword, 'ON', immediately following the REXX condition keyword, OFF, ERROR, FAILURE or HALT.

PTE60008 - SIGNAL EVENT IS NOT RECOGNIZED

A CA Enterprise Tester SIGNAL command was processed that did not specify a valid CA Enterprise Tester condition name immediately following the required 'ON' keyword. The value specified following ON must be one of the CA Enterprise Tester condition names, LOCKOUT, TIMEOUT, ENDQUEUE, LOGOFF, ENDTRACE, CANCEL or FAULT.

PTE60009 - UNKNOWN PARAMETER FOUND AFTER SIGNAL EVENT

An unknown value was found following an otherwise valid CA Enterprise Tester SIGNAL command. The format of a SIGNAL command is 'SIGNAL REXX-cond ON CA Enterprise Tester-cond'. No values may be coded following the CA Enterprise Tester-cond.

PTE60010 - QUERY RESOURCE NAME MISSING OR INVALID

The first parameter following the CA Enterprise Tester QUERY command name was not one of the keywords, QUEUE, SESSION, TRACE, TASK, QUEUES, SESSIONS, TRACES or TASKS. Correct the QUERY command by specifying the desired resource name.

PTE60011 - UNKNOWN PARAMETER FOLLOWS QUERY RESOURCE NAME

An unknown value was found following the resource name specified by a CA Enterprise Tester QUERY command. Only the QUERY QUEUE command may specify additional command parameters following the QUEUE keyword.

PTE60012 - QUERY QUEUE ARGUMENT IS NOT A REXX SYMBOL

A command parameter found following a QUERY QUEUE command is not a valid REXX variable field name. When coded within a CA Enterprise Tester host command string, REXX variable names must be distinguished from other command parameters by prefixing the variable name with an ampersand (ie., &variable.name). Only valid REXX variable names in which queued values are to be stored, may be coded following the QUERY QUEUE command and resource keywords.

PTE60013 - QUERY QUEUE MAY SPECIFY 20 SYMBOLS ONLY

More than 20 REXX variable names were specified by a CA Enterprise Tester QUERY QUEUE command. The parameter lists specified on CA Enterprise Tester queue and task management commands observe the 20 item parameter list limit of the REXX CALL instruction. In particular, no more than 20 valid REXX variable names in which queued parameter values are to be stored, may be coded following the QUERY QUEUE command and resource keywords.

PTE60014 - SESSION ALREADY ACTIVE ON ACCESSED PORT

A CA Enterprise Tester LOGON command was executed while accessing a port on which a session was already active. Only one session may be started on a particular CA Enterprise Tester port. If an additional session is needed, the ACCESS command may be used to access another port on which an additional session may be started.

PTE60015 - A LOGON PARAMETER WAS NOT RECOGNIZED

A LOGON command parameter was found that was not a LOGON command keyword or a variable field used to express a terminal screen size or terminal type. Refer to "LOGON" in topic 11.11 for a complete list of parameters supported by the CA Enterprise Tester LOGON command.

PTE60016 - CONFLICTING OR INVALID LOGON DATA FOUND

A LOGON command specified the DATA keyword, Either LOGON DATA was specified more than one time, or the logon data specified following the DATA keyword was not a valid literal value or a REXX variable field identified by a preceding ampersand (ie., &variable), that contained valid LOGON DATA. LOGON DATA is limited to 255 characters of data.

PTE60017 - USERMOD IS NOT A 1 CHARACTER OR 2 HEX CHARACTER LITERAL

The operand of the USERMOD keyword of a CA Enterprise Tester LOGON command was not a one or two digit literal value. Either a single character or a two digit hexadecimal value may be specified. The value is the character specified using the USERMOD keyword in an ISPF/PDF panel definition. The USERMOD value identifies the character to be used by ISPF/PDF to identify fields within a panel dynamic area that were modified by a terminal user.

PTE60018 - ATTRIBUTES IS NOT A 64 HEX CHARACTER LITERAL

The operand of the ATTRIBUTES keyword of a CA Enterprise Tester LOGON command was not a 64-digit hexadecimal string, expressed as a literal value, to be used as the 32 possible 3270 basic data stream field attributes when the content of a virtual terminal screen is stored in REXX variable, PTEBUFF, following the execution of each CA Enterprise Tester session command. The 32 one-byte hexadecimal values correspond to the 32 basic 3270 attributes documented in "3270 Attribute Values" in topic 14.1.

PTE60020 - TERMINAL TYPE SPECIFIED MORE THAN ONCE

A CA Enterprise Tester LOGON command specified more than one of the keywords, BASICDS or EXTDS, or specified one of the keywords redundantly. Specify no more than one of the keywords BASICDS or EXTDS to specify the CA Enterprise Tester virtual terminal type to be used for a session.

PTE60021 - MORE THAN TWO SCREEN SIZES SPECIFIED

A CA Enterprise Tester LOGON command specified more than two of the virtual terminal screen size keywords, 24X80, 32X80, 43X80 or 27X132. The first size keyword specified is taken to be the desired default virtual terminal screen size and the second is taken to be the desired alternate screen size. No more than two screen sizes may be associated with a session.

PTE60022 - INVALID VARIABLE LOGON KEYWORD FOUND

The LOGON command keywords, 24X80, 32X80, 43X80, 27X132, BASICDS or EXTDS, may be specified via a REXX variable field that is prefixed by an ampersand (ie., &default-size). A variable field was encountered while parsing a LOGON command, that was not the operand of a known LOGON command keyword, however the value of the variable field was not null or one of the supported keywords.

PTE60023 - UNABLE TO TYPE BECAUSE SESSION NOT ACTIVE ON PORT

A CA Enterprise Tester TYPE command could not be processed because a session was not active on the port being accessed by the exec. The TYPE command is valid after a session is started using a LOGON command, until the session has ended.

PTE60024 - UNABLE TO TYPE BECAUSE KEYBOARD IS NOT FREE

A CA Enterprise Tester TYPE command could not be processed because the keyboard of the virtual terminal connected to the port being accessed by the exec was not free. Variable field, PTEKB contains a value that indicates the state of the keyboard and PTESTATE contains the current state of the session. CA Enterprise Tester session commands normally complete only when the virtual terminal keyboard is free. However, CA Enterprise Tester session commands may complete while the keyboard is inhibited when a WAIT command has been used to limit the wait for a response or when other commands have been used to obtain control of special session states.

PTE60025 - A TYPE PARAMETER WAS NOT RECOGNIZED

A CA Enterprise Tester TYPE command parameter was found that was not a TYPE command keyword, a literal or variable string of TYPE data and was not a cursor positioning order. Refer to "TYPE" in topic 11.20 for a complete description of supported TYPE command parameters.

PTE60026 - A TYPE PARAMETER WAS FOUND FOLLOWING AID KEY

A TYPE command parameter was found following an AID keyword (ENTER, PF1, PA1, etc.). TYPE command data is processed in the order coded and may not be entered after an AID key is pressed because the virtual terminal keyboard is inhibited when the AID key is pressed.

PTE60028 - INVALID VARIABLE TYPE DATA FOUND

A REXX variable field, prefixed by an ampersand (ie., &variable), could not be fetched. If other messages issued by REXX or other system component do not explain the problem, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE60029 - CURSOR ROW OR POSITION IS INVALID

In a CA Enterprise Tester TYPE command, an expression of the form, < position >, or, < row column >, may be used to position the cursor within a CA Enterprise Tester virtual terminal screen. In the failing TYPE command, either position or row (the first value following '<'), was not a number or variable field, or the value of a position was zero or greater than screen size, or the value of row was zero or greater than the number of screen rows.

PTE60030 - CURSOR COLUMN IS INVALID

In a CA Enterprise Tester TYPE command, an expression of the form, < row column >, may be used to position the cursor within a CA Enterprise Tester virtual terminal screen. In the failing TYPE command, either column (the 2nd value following '<'), was not a number or variable field, or the value of column was zero or greater than the number of virtual terminal screen columns.

PTE60031 - CURSOR EXPRESSION DELIMITER MISSING

In a CA Enterprise Tester TYPE command, an expression of the form, < position >, or, < row column >, may be used to position the cursor within a CA Enterprise Tester virtual terminal screen. In the failing TYPE command, the closing '>' could not be located following position or column.

PTE60032 - ATTRIBUTE MAY NOT BE ERASED AT <r c>

A TYPE command ERASE keyword was processed while the cursor was positioned at <r c>, of the CA Enterprise Tester virtual terminal screen. The ERASE function could not be performed because the screen location is a 3270 field attribute.

PTE60033 - PROTECTED DATA MAY NOT BE ERASED AT <r c>

A TYPE command ERASE keyword was processed while the cursor was positioned at <r c>, of the CA Enterprise Tester virtual terminal screen. The ERASE function could not be performed because the screen location is in a 3270 protected screen field.

PTE60034 - USERMOD NOT OPERATIVE - PTEBUFF NOT STORED

A CA Enterprise Tester TYPE command specified the USERMOD keyword. The USERMOD keyword causes the REXX variable, PTEBUFF, or 'PTEBUFF.port' if port has been explicitly accessed, to be fetched from the REXX variable pool, and causes fields contained in the buffer that are flagged by the USERMOD character specified on the LOGON command that was used to start the session, to be keyed into the accessed virtual terminal at corresponding screen locations. The USERMOD operation could not be performed because PTEBUFF or PTEBUFF.port could not be fetched.

PTE60035 - USERMOD FIELD DOES NOT CORRESPOND TO MODIFIABLE SCREEN FIELD

A CA Enterprise Tester TYPE command specified the USERMOD keyword. The USERMOD keyword causes the REXX variable, PTEBUFF, or 'PTEBUFF.port' if port has been explicitly accessed, to be fetched from the REXX variable pool, and causes fields contained in the buffer that are flagged by the USERMOD character specified on the LOGON command that was used to start the session, to be keyed into the accessed virtual terminal at corresponding screen locations. The USERMOD operation could not be completed successfully because usermod data in the fetched buffer does not correspond to a modifiable field in the accessed virtual terminal screen buffer.

PTE60036 - USERMOD FIELD CONTAINS INVALID DATA (HEX 01-3F)

A CA Enterprise Tester TYPE command specified the USERMOD keyword. The USERMOD keyword causes the REXX variable, PTEBUFF, or 'PTEBUFF.port' if port has been explicitly accessed, to be fetched from the REXX variable pool, and causes fields contained in the buffer that are flagged by the USERMOD character specified on the LOGON command that was used to start the session, to be keyed into the accessed virtual terminal at corresponding screen locations. A modified field in PTEBUFF was found to contain data in the range hex 01 through hex 3f, that does not correspond to attribute locations in the virtual terminal screen buffer. A basic data stream virtual terminal is being used for the session.

PTE60037 - USERMOD FIELD IS ZERO LENGTH

A CA Enterprise Tester TYPE command specified the USERMOD keyword. The USERMOD keyword causes the REXX variable, PTEBUFF, or 'PTEBUFF.port' if port has been explicitly accessed, to be fetched from the REXX variable pool, and causes fields contained in the buffer that are flagged by the USERMOD character specified on the LOGON command that was used to start the session, to be keyed into the accessed virtual terminal at corresponding screen locations. A modified field in PTEBUFF was found, however the buffer position immediately following the occurrence of the USERMOD character was also a USERMOD character. A zero length screen field can not be modified by a terminal user.

PTE60038 - USERMOD INTERNAL LOGIC ERROR

A CA Enterprise Tester TYPE command specified the USERMOD keyword. The USERMOD keyword causes the REXX variable, PTEBUFF, or 'PTEBUFF.port' if port has been explicitly accessed, to be fetched from the REXX variable pool, and causes fields contained in the buffer that are flagged by the USERMOD character specified on the LOGON command that was used to start the session, to be keyed into the accessed virtual terminal at corresponding screen locations. A probable logic error was detected while performing the USERMOD operation. Contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE60042 - TYPE DATA MAY NOT INCLUDE ATTRIBUTE VALUE, HEX xx

The character, hexadecimal 'xx', was found in a string of data specified in a CA Enterprise Tester TYPE command. The character can not be keyed into the basic 3270 data stream terminal being used for the session because the value is used to represent a 3270 basic field attribute. Only an online system may place field attributes in the virtual terminal buffer. The value may be a keyable character if a 3270 extended data stream virtual terminal is used for a session. An extended data stream terminal is selected by specifying EXTDS on the LOGON command that is used to start a session.

PTE60043 - "x" MAY NOT BE KEYED OVER ATTRIBUTE AT <r c>

The character, "x", was found in a string of data specified in a CA Enterprise Tester TYPE command. The character can not be keyed into the virtual terminal screen buffer at screen row and column '<r c>', because that screen location is a 3270 field attribute. The TYPE command can key data into unprotected screen locations only.

PTE60044 - ProTerm SYSTEM ERROR - ATTRIBUTE NOT FOUND

While processing a CA Enterprise Tester TYPE command, an attempt was made to locate an attribute within a formatted virtual terminal screen buffer. The attribute could not be located. This error should never occur. Contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE60045 - "x" MAY NOT BE KEYED AT PROTECTED LOCATION <r c>

The character, "x", was found in a string of data specified in a CA Enterprise Tester TYPE command. The character can not be keyed into the virtual terminal screen buffer at screen row and column '<r c>', because that screen location is in a 3270 protected field. The TYPE command can key data into unprotected screen locations only.

PTE60046 - TYPE CHARACTER HEX xx MAY NOT BE KEYED

The character, hexadecimal 'xx', was found in a string of data specified in a CA Enterprise Tester TYPE command. The values hex 01 through 3F, with the exception of FM and DUP characters are reserved by 3270 architecture for use as control characters and may not be keyed into a terminal.

PTE60048 - INVITE INTERVAL NOT HHMMSSTH

A value was found in an INVITE command that was not a valid time interval. A time interval may be expressed in the format hhmmssth, where hh is hours, mm is minutes, ss is seconds and th is tenths and hundredths of seconds, or may be expressed using a variable field, prefixed by an ampersand (ie., &invtime), that contains a valid time interval.

PTE60049 - UNKNOWN PARM FOLLOWS INVITE TIME

A CA Enterprise Tester INVITE command was processed that specified a valid time interval, however an additional parameter was found following the time expression. The INVITE command supports only a single parameter and additional parameters are not accepted.

PTE60050 - TERMINAL TASK ENDED DURING INVITE

The z/OS subtask used to manage the session started by a CA Enterprise Tester LOGON command unexpectedly ended while waiting for a message during INVITE command processing. This error indicates a system or CA Enterprise Tester product problem. If other messages issued to the job log, system log, or elsewhere do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE60051 - AN UNKNOWN LOGOFF PARM WAS FOUND

An unknown value was found following a LOGOFF command. The CA Enterprise Tester LOGOFF command ends any session on the port being accessed by an exec and supports no parameters. No value may be specified following the LOGOFF command name.

PTE60052 - A TRACE PARAMETER IS INVALID

While processing a CA Enterprise Tester TRACE command, a parameter was found that was not a supported TRACE keyword or a literal or variable TRACE caption. Refer to "TRACE" in topic 11.19 for a list of supported TRACE command parameters.

PTE60053 - A TRACE PARAMETER IS CONFLICTING OR REDUNDANT

A TRACE command specified a redundant or conflicting parameter. Either the TO keyword was specified twice, or more than one of the keywords, IMAGE, FORMAT or UPPERCASE were specified, or more than one of the keywords, PACK or NOPACK was specified, or more than one TRACE caption was specified.

PTE60054 - TRACE CAPTION EXCEEDS 40 CHARACTERS

A TRACE command specified a trace caption to be written with subsequently traced screen images. The caption may have been expressed as a literal or variable value. The caption specified is invalid because it exceeds the 40 character caption limit.

PTE60055 - TRACE ALREADY ACTIVE ON ACCESSED PORT

A TRACE command was processed that specified the TO keyword and a DD name to which session screens were to be traced. The trace could not be started because a trace was already active on the port being accessed by the exec. Only one session trace may be started on a port.

PTE60057 - TRACE PARAMETER(S) NOT SPECIFIED

A TRACE command was processed that specified no trace parameters. Every TRACE command must specify at least one parameter.

PTE60058 - TRACE NOT ACTIVE ON ACCESSED PORT

A TRACE command was processed that specified only a trace format, FORMAT, IMAGE or UPPERCASE, or that specified a trace caption. The TRACE command did not specify the TO keyword and a DD name to which traced screens were to be written, nor was a trace active at the time that the command was processed. A TRACE command that is used to change the trace caption or format only may be issued only while a trace is active.

PTE60059 - VARIABLE TRACE CAPTION INVALID

A TRACE command contained a REXX variable name (prefixed with an ampersand, ie., &caption), that was taken to be a variable trace caption. The caption could not be fetch from the REXX variable pool, possibly because the caption value exceeded the 40 character limit for captions. If the length of the variable does not exceed 40 characters, a system error has occurred, in which case you CA Enterprise Tester technical support may be of assistance in resolving the problem.

PTE60060 - SPECIFY PACK/NOPACK WITH TO DD ONLY

A TRACE command specified either the PACK or NOPACK command. However, the TRACE command did not also specify the TO keyword and a DD name to instruct CA Enterprise Tester to open the data set to which subsequently traced screen images were to be written. The PACK or NOPACK keyword may be specified only on a TRACE command that is used to open a trace data set.

PTE60061 - AN UNKNOWN ENDTRACE PARAMETER WAS FOUND

An unknown value was found following an ENDTRACE command. The CA Enterprise Tester ENDTRACE command ends a session trace on the port being accessed by an exec and supports no parameters. No value may be specified following the ENDTRACE command name.

PTE60062 - EXEC SUBTASK ABEND Sxxx Uxxx

While processing a CA Enterprise Tester ATTACH command, the CA Enterprise Tester exec subtask program, PTEXS, was attached using the z/OS ATTACH macro. The CA Enterprise Tester exec subtask creates a REXX Language Processor Environment and runs a REXX exec named by the CA Enterprise Tester ATTACH command. The exec subtask program abended. The message documents the system and/or user abend code incurred by the exec subtask program. Refer to IBM publications that document system abend codes. If the abend codes or other system messages issued during abend processing do not explain the cause of the error, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE60063 - PTEXS ATTACH FAILED, RC=rc

While processing a CA Enterprise Tester ATTACH command, the CA Enterprise Tester exec subtask program, PTEXS, was attached using the z/OS ATTACH macro. The CA Enterprise Tester exec subtask creates a REXX Language Processor Environment and runs a REXX exec named by the CA Enterprise Tester ATTACH command. The z/OS ATTACH macro completed with a non-zero return code. This message documents the z/OS ATTACH macro return code, 'rc'. Refer to IBM publications that document the z/OS ATTACH macro. If the return code or other system messages issued at the time of the error do not explain the cause of the error, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE60064 - UNKNOWN IRXINIT RC=rc

The CA Enterprise Tester exec subtask program, PTEXS, called REXX program, IRXINIT, to initialize a REXX language processor environment in which to run an exec named by a CA Enterprise Tester ATTACH command. The meaning of the return code from IRXINIT, 'rc', is unknown to CA Enterprise Tester. Refer to the IBM publication, 'TSO/E, REXX REFERENCE', for information about IRXINIT and IRXINIT return codes. If the cause of the problem can not be determined based on the return code and other system messages that may have been issued at the time of the error, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE60065 - IRXINIT FAILED - RC=20, REASON=reason

The CA Enterprise Tester exec subtask program, PTEXS, called REXX program, IRXINIT, to initialize a REXX language processor environment in which to run an exec named by a CA Enterprise Tester ATTACH command. For RC=20, IRXINIT indicates the cause of the failure using a 'reason' code. Refer to the IBM publication, 'TSO/E, REXX REFERENCE', for information about IRXINIT and IRXINIT reason codes. If the cause of the problem can not be determined based on the reason code and other system messages that may have been issued at the time of the error, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE60066 - IRXINIT ABENDED Sxxx/Uxxx, REASON xxxx

The CA Enterprise Tester exec subtask program, PTEXS, called REXX program, IRXINIT, to initialize a REXX language processor environment in which to run an exec named by a CA Enterprise Tester ATTACH command. IRXINIT incurred a system abend, Sxxx, or a user abend, Uxxx, while initializing the environment. The abend reason code, xxxx, is also shown in the message. Refer to the IBM publication, 'TSO/E, REXX REFERENCE', for information about IRXINIT abend handling. If the cause of the problem can not be determined based on the abend code and other system messages that may have been issued at the time of the error, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE60067 - IRXSUBCM UPDATE FAILED - RC=rc

After using IRXINIT to initialize a REXX language processor environment in which to run an exec named in a CA Enterprise Tester ATTACH command, the CA Enterprise Tester exec subtask program called REXX program, IRXSUBCM, to update the CA Enterprise Tester entry in the REXX Host Command Environment Table. The table update request completed with a non-zero return code, 'rc'. Refer to the IBM publication, 'TSO/E, REXX REFERENCE', for information about IRXSUBCM return codes. If the cause of the problem can not be determined based on the return code and other system messages that may have been issued at the time of the error, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE60068 - EXEC NOT FOUND - execname

After using IRXINIT and IRXSUBCM to initialize and prepare a REXX language processor environment, the CA Enterprise Tester exec subtask program called REXX program, IRXLOAD, to load the exec named in the CA Enterprise Tester ATTACH command, 'execname'. The REXX routine completed with a return code of -3, indicating that the exec could not be located. Probable causes of the error are that the named exec is not in an exec library that is accessible to the REXX load routine, or that the named exec was misspelled.

PTE60069 - EXEC LOAD FAILED - IRXLOAD RC=rc

After using IRXINIT and IRXSUBCM to initialize and prepare a REXX language processor environment, the CA Enterprise Tester exec subtask program called REXX program, IRXLOAD, to load the exec named in the CA Enterprise Tester ATTACH command. The REXX load routine completed with a non-zero return code, 'rc'. Refer to the IBM publication, 'TSO/E, REXX REFERENCE', for information about IRXLOAD return codes. If the cause of the problem can not be determined based on the return code and other system messages that may have been issued at the time of the error, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE60070 - ATTACH FAILED BECAUSE EXEC ACTIVE ON PORT

A CA Enterprise Tester ATTACH command was executed while accessing a port on which an exec was already running. Only one exec may be running on a particular CA Enterprise Tester port at a time. The ACCESS command may be used to access a port on which no exec is running prior to issuing the ATTACH command.

PTE60071 - ATTACH DOES NOT SPECIFY TASK TYPE

An ATTACH command was processed that did not identify the type of task to be attached. Currently, the only task type supported by the ATTACH command is a REXX exec task. The keyword, EXEC, must be coded immediately following the ATTACH command name and immediately before the REXX exec name, to identify the task to be attached as a REXX subtask.

PTE60072 - UNKNOWN CANCEL PARAMETER FOUND

An unknown value was found following a CANCEL command. The CA Enterprise Tester CANCEL command cancels any exec on the port being accessed by an exec and supports no parameters. No value may be specified following the CANCEL command name.

PTE60073 - A PULL ARGUMENT IS NOT A VALID REXX SYMBOL

A CA Enterprise Tester PULL command contained an argument that was not a valid REXX symbol. When REXX symbols are specified in command text that is seen by CA Enterprise Tester, an ampersand, must immediately precede the symbol, to distinguish the symbol from current and future CA Enterprise Tester keywords. A PULL command may specify from 0 through 20 symbols, in which corresponding values obtained from a pulled CA Enterprise Tester queue entry are to be stored. No other PULL command parameters are supported.

PTE60074 - PULL COMMAND MAY NOT SPECIFY MORE THAN 20 REXX SYMBOLS

A CA Enterprise Tester PULL command listed more than 20 REXX variable names. When a PULL command takes an entry out of a CA Enterprise Tester queue, data values contained in the queue entry are stored in corresponding PULL command variable fields. The failed CA Enterprise Tester command listed more than 20 literal or variable parameter values. The limit of 20 parameters parallels the 20 parameter limit of the REXX CALL instruction.

PTE60075 - AN UNSUPPORTED DELETEQ PARAMETER WAS FOUND

An unknown value was found following a DELETEQ command. The CA Enterprise Tester DELETEQ command deletes any queued records on the port being accessed by an exec and supports no parameters. No value may be specified following the DELETEQ command name.

PTE60076 - IRXEXCOM STORE FAILED, RC=xx

An attempt to store a REXX variable field value failed. The return code, expressed in hexadecimal, returned by the REXX variable access routine, IRXEXCOM, replaces 'rc' in the message. The IBM publication, 'TSO/E, REXX Reference' (SC28-1883-x) documents IRXEXCOM return codes. However, this error should never occur. If the cause of the problem can not be determined, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE60077 - WAIT TIME IS NOT HHMMSSTH OR FOREVER

The first parameter specified in a WAIT command was not a valid time interval or the keyword FOREVER. A time interval may be expressed in the format hhmmssth, where hh is hours, mm is minutes, ss is seconds and th is tenths and hundredths of seconds. A variable field, prefixed by an ampersand (ie., &waittime), that contains a valid time or the keyword, FOREVER, may be used to specify a valid WAIT interval.

PTE60078 - VALUE AFTER WAIT INTERVAL IS NOT ON

While processing a CA Enterprise Tester WAIT command, a value was found following the WAIT interval (or FOREVER keyword), however the value was not the keyword, 'ON'. When a WAIT command is used to place a limit on the time that CA Enterprise Tester will wait for the completion of a named event, the keyword, ON, must immediately follow the time interval and must immediately precede the name of the event to which the time limit is to be applied. For example, 'WAIT 1000 ON RESPONSE'.

PTE60079 - WAIT FOREVER WITHOUT ON IS INVALID

A WAIT command was processed that specified only the FOREVER keyword. A WAIT command may specify the FOREVER keyword only when the ON keyword is also specified to place a limit on the wait for an event whose name follows the ON keyword..

PTE60080 - UNKNOWN VALUE FOUND AFTER WAIT ON

A WAIT command that specified the ON keyword specified a value following ON that was not one of the supported event name keywords, LOCK, PULL or RESPONSE. When ON is specified in a WAIT command, one of the supported event name keywords must be specified immediately following ON.

PTE60081 - UNKNOWN VALUE FOUND AFTER WAIT EVENT NAME

An unknown value was found following a WAIT command event name that followed the ON keyword. When a WAIT command is used to place a limit on the time CA Enterprise Tester will wait for the completion of a named event, the event name must be the last value specified by the WAIT command.

PTE60084 - CONFLICTING resource-name FOUND

A resource that should be named no more than once in a CA Enterprise Tester command was specified more than once. When the message is stored in PTEMESS, 'resource-name' is replaced by the name of the conflicting resource. For example, if terminal model were specified twice in a LOGON LOGON command, resource name would be replaced by 'TERMINAL MODEL'.

PTE60085 - INVALID resource-name SPECIFIED

A resource name specified in a CA Enterprise Tester command was not valid. When the message is stored in PTEMESS, 'resource-name' is replaced by the name of the resource for which an invalid value was specified. For example, if the name of a REXX exec specified by an attach command is invalid, then resource-name would be replaced by 'REXX EXEC NAME'.

PTE60086 - REQUIRED resource-name NOT FOUND

A resource name that is required for a processed CA Enterprise Tester command was not specified. When the message is stored in PTEMESS, 'resource-name' is replaced by the name of the resource for which a value must be specified. For example, if a LOGON command did not specify an online system name, then resource-name would be replaced by 'SYSTEM NAME'.

PTE60087 - INVALID VARIABLE NUMBER FOUND

A REXX variable field was used in an ACCESS command as a port number, or in a TYPE command as a screen position, row or column number, however, the variable field value is not an integer.

PTE60088 - A LISTED PARAMETER IS NOT A LITERAL OR A SYMBOL

A CA Enterprise Tester ATTACH, QUEUE, PUSH or POST command specified an invalid parameter value. Each of these commands may specify a list of 0 through 20 data items to be passed as parameters to an attached exec or to be stored as data items in a single CA Enterprise Tester queue entry. The data item list may contain literal values, enclosed in single or double quotes (as seen by CA Enterprise Tester), or may contain REXX variable names. When REXX variable names are specified in command text that is seen by CA Enterprise Tester, an ampersand, must immediately precede the name to distinguish the name from current and future CA Enterprise Tester keywords. The failed CA Enterprise Tester command listed a value that was not a literal value or a REXX symbol that contains a value.

PTE60089 - PARAMETER LIST CONTAINS MORE THAN 20 PARAMETER VALUES

A CA Enterprise Tester ATTACH, QUEUE, PUSH or POST command listed more than 20 data parameter values. Each of these commands may specify a list of 0 through 20 data items to be passed as parameters to an attached exec or to be stored as data items in a single CA Enterprise Tester queue entry. The failed CA Enterprise Tester command listed more than 20 literal or variable parameter values. The limit of 20 parameters parallels the 20 parameter limit of the REXX CALL instruction.

PTE60090 - "c", HEX xx, IS AN UNSUPPORTED CHARACTER

While parsing a CA Enterprise Tester command, an unsupported character was encountered. When the command is stored in PTEMESS, the unsupported is inserted into the message in character and hexadecimal format.

PTE60091 - LITERAL CLOSING QUOTE NOT FOUND

While processing a CA Enterprise Tester command, a literal was found, however the quote that marks the end of the literal value was not contained in the command.

PTE60092 - NUMBER MAY NOT EXCEED 15 DIGITS

While parsing a CA Enterprise Tester command, a number was encountered that was larger than 15 digits. No parameter of any CA Enterprise Tester command has a use for a 15 digit number and the number is therefore excessive.

PTE60093 - VARIABLE NAME IS INVALID

An ampersand was found while parsing a CA Enterprise Tester command. An ampersand is used in the text of CA Enterprise Tester commands to identify the following character as the beginning of a REXX variable name. The variable name is invalid because the character that follows the ampersand is not a character that may validly be the first character of a REXX symbol.

PTE60094 - HANDLE CONDITION IS NOT SUPPORTED

A CA Enterprise Tester HANDLE command was processed. The first word following HANDLE was not an event or condition name that is supported by the HANDLE command. Supported event or condition names are TURNAROUND or OUTBOUND.

PTE60095 - HANDLE OPTION IS NOT ON OR OFF

A CA Enterprise Tester HANDLE command was processed. The word following the event or condition name for which handling was being specified was not 'ON' or 'OFF'. Automatic handling of a named event or condition may be activated by specifying 'ON'. Automatic handling of a named event or condition may be inactivated by specifying 'OFF'. No other HANDLE command options are supported.

PTE60096 - UNKNOWN VALUE FOLLOWS HANDLE OPTION

A CA Enterprise Tester HANDLE command was processed. Following the HANDLE option, ON or OFF, another value was found. The HANDLE command may specify the name of an event or condition and one of the options ON or OFF, in that order and may specify no additional parameters.

PTE60100 - MONITOR COMMAND PARMS ARE REDUNDANT CONFLICTING OR INVALID

A invalid CA Enterprise Tester MONITOR command was executed. A MONITOR command is invalid if (1) it specifies a MONITOR command keyword redundantly, (2) specifies more than one of the keywords NOTE, RECALL, VIA or OFF, (3) specifies NOTE or RECALL and the operand of the keyword is not a rexx symbol preceded by an ampersand, (4) specifies no MONITOR options, or (5) if an error occurs while fetching a noted value during the execution of a MONITOR RECALL command.

PTE60101 - SELECTED SCREEN LOCATION IS NOT PEN DETECTABLE <r c>

A CA Enterprise Tester TYPE command SELECT keyword was executed while the virtual terminal cursor was not positioned in a pen detectable field. with a valid designator character in position 1 of the field. When the message is issued, <r c>, is replaced by a similar expression that includes the cursor row and column at the time the SELECT keyword was executed. To perform operations on a pen detectable field similar to those performed by the cursor select key or a light pen, first position the cursor somewhere within a pen detectable field. The pen detectable field may be either a selectable field or a pen attention field.

PTE60102 - DESELECTED SCREEN LOCATION IS NOT PEN DETECTABLE <r c>

A CA Enterprise Tester TYPE command DESELECT keyword was executed while the virtual terminal cursor was not positioned in a pen detectable field with a valid field selection designator character, ? or >. When the message is issued, <r c>, is replaced by a similar expression that includes the cursor row and column at the time the DESELECT keyword was executed. To deselect a pen selectable field first position the cursor somewhere within the pen selectable field.

PTE60103 - DELAY MUST SPECIFY TIME OR PERCENT

A CA Enterprise Tester DELAY command was not executed because the first command argument following the DELAY keyword was not a valid DELAY time or the keyword, PERCENT. DELAY time may be expressed as numeric digits in the form, hhmmssth, where hh is hours, mm is minutes, ss is seconds, and th is tenths and hundredths of a second, or may be a variable field that contains DELAY time in the form, hhmmssth. A DELAY statement must specify either a time or the PERCENT keyword.

PTE60104 - DELAY TIME NOT HHMMSSTH

A CA Enterprise Tester DELAY command was executed that specified an apparent delay time as a string of numeric digits or as a variable field (rexx symbol preceded by an ampersand) that contained a delay time. However, the numeric digits or the variable field value was not a valid delay time expressed in the form, hhmmssth, where hh is hours, mm is minutes, ss is seconds, and th is tenths and hundredths of a second.

PTE60105 - DELAY PERCENT NOT 0-999

A CA Enterprise Tester DELAY command was executed that specified the PERCENT keyword, however the value following the PERCENT keyword was not a valid percent. A valid DELAY PERCENT is an integer from 0 through 999 and may be expressed as an actual integer or as a variable field that contains a valid delay percent. Specify a percent from 0 through 99 to reduce DELAY command times and make an exec session run faster, or specify a percent from 101 to 999 to extend DELAY command times and to make an exec session run slower.

PTE60106 - UNKNOWN DELAY PARAMETER FOUND

A DELAY command was executed that specified an apparently valid delay time or percent, however an unknown argument was found following the specified delay time or percent. A DELAY COMMAND may specify only a valid delay time or the PERCENT keyword followed by a valid delay percent, and any other argument specified on a DELAY command is invalid.

PTE60107 - ATTACH SAYDD ALLOCATION FAILED

During execution of a CA Enterprise Tester ATTACH command, it was found that the default REXX SAY destination or SAYDD specified by the ATTACH command was not allocated. The ATTACH command attempted to allocate a data set to the default or specified ddname using any specified SAYDEST or default destination specified by the CA Enterprise Tester Options Table PTEOM REXX SAYDD and/or SAYDEST specifications. The allocation was not successful. If invalid ATTACH command SAYDD or SAYDEST options were specified, correct the ATTACH command or verify that PTEOM REXX SAYDD and SAYDEST specifications for the address space in which the ATTACH command was executed are valid.

PTE60108 - A SESSION TIMEOUT OCCURRED

A session timeout occurred while waiting on a CA Enterprise Tester session event, during the execution of a CA Enterprise Tester LOGON, INVITE, TYPE, ATTN or DELAY host command. A session timeout occurs when a single session event does not complete within the time interval specified on a WAIT command for RESPONSE. When a LOGON command times out, the partially initialized session is terminated. When a timeout occurs during a TYPE, ATTN, INVITE or DELAY command, the session remains active, and the exec may issue an ATTN command to send an expedited SNA signal or may issue an INVITE or DELAY command to resume waiting for the completion of the timed-out session event.

PTE60109 - PTETS SESSION/TRACE SUBTASK FAILED

An z/OS subtask used to manage a session and related resources on a port unexpectedly ended while processing a CA Enterprise Tester LOGON, TYPE, ATTN, INVITE, DELAY, LOGOFF, TRACE or ENDTRACE host command. This error indicates a system or CA Enterprise Tester product problem. If other messages issued to the job log, system log, or elsewhere do not explain the cause of the problem, contact CA Enterprise Tester technical support for assistance.

PTE61001 - UNKNOWN PROTERM HOST COMMAND

The first word of the command string provided to the CA Enterprise Tester host command processor was not a supported CA Enterprise Tester host command. Either address the correct REXX host command environment or correct the CA Enterprise Tester host command name.

PTE61002 - HOST COMMAND TOKEN INIT FAILED

While processing the first CA Enterprise Tester Host command in an address space, an attempt to update the CA Enterprise Tester "host command token" failed. The host command token is used by CA Enterprise Tester to keep track of resources created and managed by CA Enterprise Tester host commands. The CA Enterprise Tester Host Command Environment is inoperative until host command token initialization is successful. Contact CA Enterprise Tester technical support for assistance if the cause of the problem can not be determined.

PTE61003 - PROTERM CONTROL TASK INIT FAILED

While processing the first CA Enterprise Tester Host command in an address space, an attempt to attach the CA Enterprise Tester host command processor control task failed. The CA Enterprise Tester control task is used to provide central services to execs running in z/OS multitasking environments. The CA Enterprise Tester Host Command Environment is inoperative until control task initialization is successful. Contact CA Enterprise Tester technical support for assistance if the cause of the problem can not be determined.

PTE61004 - PROTERM CONTROL TASK INIT ABENDED

While processing the first CA Enterprise Tester Host command in an address space, the CA Enterprise Tester host command processor control task was successfully attached, however, the control task abended during initial subtask processing. The CA Enterprise Tester control task is used to provide central services to execs running in z/OS multitasking environments. The CA Enterprise Tester Host Command Environment is inoperative until control task initialization is successful. Contact CA Enterprise Tester technical support for assistance if the cause of the problem can not be determined.

PTE61005 - PROTERM SIGNALED signal ON condition

During the execution of a CA Enterprise Tester Host Command, when a CA Enterprise Tester condition is encountered or a REXX signal is raised, and no other specific is applicable, PTE61005, is raised to document the CA Enterprise Tester condition and/or REXX signal raised. In the message, 'signal' is replaced by one of the REXX signal keywords, HALT, FAILURE, ERROR or RC, and 'condition', is replaced by one of the CA Enterprise Tester condition keywords, FAULT, CANCEL, ENDTRACE, LOGOFF, ENDQUEUE, TIMEOUT, LOCKOUT, or RC. The message is information only and does not constitute an error unless the condition (also expressed as a REXX RC), or the REXX signal is considered to be an error by the REXX application.

PTE62001 - INVALID IRXEFPL AT ENTRY TO PTERI

The REXX External Function Parameter List provided to CA Enterprise Tester function PTERI, did not conform to REXX specifications or CA Enterprise Tester requirements. PTERI accepts one and only one parameter. The parameter specified must be one of the CA Enterprise Tester function names, START or END.

PTE62002 - UNKNOWN PTERI COMMAND

The CA Enterprise Tester function, PTERI, was invoked or called, however the parameter specified was not one of the CA Enterprise Tester function names, START or END. PTERI(START) may be used to dynamically initialize the CA Enterprise Tester Host Command Environment prior to proper installation. PTERI(END) may be used to perform appropriate exec-termination-like cleanup prior to proper installation of the CA Enterprise Tester Host Command Environment.

PTE62003 - UNKNOWN START COMMAND PARAMETER FOUND

An unknown value was found following the PTERI START command. When the START command is specified as the first parameter to the CA Enterprise Tester PTERI function, only the START command may be specified. The CA Enterprise Tester Host Command Environment is not initialized.

PTE62004 - PROTERM HOST COMMAND TOKEN WAS RESET

The CA Enterprise Tester PTERI START function detected a partially initialized, but corrupted CA Enterprise Tester Host Command Environment control block. The CA Enterprise Tester Host command token was reset to uninitialized state. If problem persists, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE62005 - UNKNOWN END COMMAND PARAMETER FOUND

An unknown value was found following the PTERI END command. When the END command is specified as the first parameter to the CA Enterprise Tester PTERI function, only the END command may be specified. The CA Enterprise Tester Host Command Environment is not terminated.

PTE63013 - PTETS MESSAGE ILLOGIC

An error message was issued by the CA Enterprise Tester terminal task, however due to a logic error in message processing, the message issued can not be displayed. Contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE64001 - dsn NOT FOUND

A data set, "dsn", specified as the target for a CA Enterprise Tester recording, trace or log contained a parenthetical expression which was assumed to be a relative generation number. An attempt to do a LOCATE to convert the relative expression to an absolute generation number failed. Either an incorrect data set name or an invalid relative generation number was specified or the relative generation data set is not cataloged.

PTE64002 - dsn IS BEING USED BY ANOTHER TASK

An attempt to obtain an exclusive ENQ on a data set, "dsn", that was specified as the target for a CA Enterprise Tester recording, trace or log, was not successful. Failure to obtain the ENQ indicates that another job, task or TSO user owns an ENQ on the specified data set or data set and member. Either specify another target for the recording, trace or log, or wait until the holder of the data set or PDS member releases control of the data.

PTE64003 - ALLOC FOR dsn FAILED, RC xxxxxxxx

An attempt to dynamically allocate a data set, "dsn", specified as a target for a CA Enterprise Tester recording, trace or log, was unsuccessful. When the message is issued, 'xxxxxxxx' is replaced by the SVC 99 error reason and information code. Refer to Appendix A of the help that is available when the HELP key is pressed while viewing the ISPF/PDF PRIMARY OPTION MENU or other IBM documentation of SVC 99 services for an explanation of the dynamic allocation error reason and information code.

PTE64004 - dsn IS NOT A PDS, MEMBER DISALLOWED

A partitioned data set member, "dsn", was specified as the target for a CA Enterprise Tester recording, trace or log. It was found that the data set specified was not a partitioned data set. Either specify a sequential data set without a member name or specify a member of a partitioned data set.

PTE64005 - dsn DSORG IS NOT PS OR PO

A data set, "dsn", specified as target for a CA Enterprise Tester recording, trace or log, is not a sequential or partitioned data set. CA Enterprise Tester recordings, traces and logs may be written to partitioned data set members and sequential data sets and SYSOUT data sets only.

PTE64006 - SPECIFY MEMBER IN dsn

A data set, "dsn", specified as a target for a CA Enterprise Tester recording, trace or log was found to be a partitioned data set. However, a member name was not specified in the RECORD, TRACE or LOG request. Specify a member name whenever data is to be captured in a PDS.

PTE64007 - UNABLE TO OBTAIN DSCB FOR dsn

An z/OS OBTAIN macro was issued to obtain the Format 1 DSCB from the VTOC of the volume containing data set, "dsn". The request failed. "dsn" was specified as a target for a CA Enterprise Tester recording, trace or log. Because the volume serial number was obtained during successful dynamic allocation of the data set, this problem should not occur. The error may indicate a problem with the VTOC of the volume containing the data set. Contact a technical support person at your installation for assistance in resolving the problem.

PTE64008 - RECFM OF dsn IS NOT VALID FOR form

The record format (RECFM) of a data set, "dsn", specified as the target for a CA Enterprise Tester recording, trace or log, is not acceptable for the type of data being captured. When the message is issued, "form", is one of the values RECORD, TRACE, or LOG, indicating the type of data being captured. CA Enterprise Tester recordings may be written to RECFM=F or RECFM=FB data sets. CA Enterprise Tester traces may be written to RECFM=F, RECFM=FB, RECFM=FA or RECFM=FBA data sets. CA Enterprise Tester logs may be written to RECFM=FA or RECFM=FBA data sets. Standard blocking may also be specified for all capture data sets.

PTE64009 - LRECL OF dsn IS NOT VALID FOR form

The logical record length (LRECL) of a data set, "dsn", specified as the target for a CA Enterprise Tester recording, trace or log is not acceptable for the type of data being captured. When the message is issued, "form", is one of the values RECORD, TRACE, or LOG, indicating the type of data being captured. CA Enterprise Tester recordings must be written to data sets with LRECL=80 only. CA Enterprise Tester traces may be written to data sets with LRECL=133 through LRECL=256 (LRECL=135 is preferred). CA Enterprise Tester logs must be written to data sets with LRECL=151 only.

PTE64010 - BLKSIZE OF dsn IS NOT VALID

The block size (BLKSIZE) of a data set, "dsn", specified as the target for a CA Enterprise Tester recording, trace or log is not acceptable. For unblocked records, BLKSIZE must equal LRECL. For blocked records BLKSIZE must be a multiple of LRECL.

PTE64011 - tddn DD NOT ALLOCATED

A ddname, "tddn", specified as the target for a CA Enterprise Tester recording, trace or log, was not allocated to the address space when CA Enterprise Tester attempted to open the data set. When a CA Enterprise Tester script is executed as a batch job step using program PTEBS, a PTELOG DD statement must be present in the job step JCL. A ddname specified using the Script Language RECORD or TRACE statement TO keyword must be allocated to the address space prior to execution of the RECORD or TRACE statement.

PTE64012 - tddn DD * IS INVALID form TARGET

A ddname, "tddn", specified as the target for a CA Enterprise Tester recording, trace or log was allocated as a SYSIN data set, and may not be written to. When the message is issued, "form", is one of the values RECORD, TRACE, or LOG, indicating the type of data being captured. CA Enterprise Tester recordings, traces and logs may be written only to sequential data sets, partitioned data set members or as SYSOUT data.

PTE64013 - dsn DSORG IS NOT PS OR PO

A data set, "dsn", specified as target for a CA Enterprise Tester recording, trace or log, is not a sequential or partitioned data set. CA Enterprise Tester recordings, traces and logs may be written to partitioned data set members and sequential data sets or SYSOUT data sets only.

PTE64014 - dsn IS A PDS, SPECIFY MEMBER

A data set, "dsn", specified as a target for a CA Enterprise Tester recording, trace or log was found to be a partitioned data set. However, a member name was not specified in the JCL DD statement or ALLOCATE statement used to allocate the data set to the address space.

PTE64015 - DISP=MOD INVALID FOR PDS dsn

A data set, "dsn", specified as a target for a CA Enterprise Tester recording, trace or log was allocated with DISP=MOD. DISP=MOD may not be used to write to a member of a partitioned data set.

PTE64016 - dsn IS BEING USED BY ANOTHER TASK

An attempt to obtain an exclusive ENQ on a data set, "dsn", that was specified as the target for a CA Enterprise Tester recording, trace or log, was not successful. Failure to obtain the ENQ indicates that another job, task or TSO user owns an ENQ on the specified data set or data set and member. Either specify another target for the recording, trace or log, or wait until the holder of the data set or PDS member releases control of the data.

PTE64017 - dsn IS NOT A PDS, MEMBER DISALLOWED

A partitioned data set member, "dsn", was specified as the target for a CA Enterprise Tester recording, trace or log. It was found that the data set specified was not a partitioned data set. Either specify a sequential data set without a member name or specify a member of a partitioned data set.

PTE64018 - RDJFCB FOR dsn FAILED

The JFCB for data set, "dsn", was read to obtain any JCL DCB information. "dsn" was specified as the target for a CA Enterprise Tester recording, trace or log. The attempt to read the JFCB was unsuccessful. Because it is known that the data set is allocated, this error should not occur. If you are unable to determine the cause of the problem, contact CA Enterprise Tester technical support for assistance.

PTE64019 - UNABLE TO OBTAIN DSCB FOR dsn

An z/OS OBTAIN macro was issued to obtain the Format 1 DSCB from the VTOC of the volume containing data set, "dsn". The request failed. "dsn" was specified as a target for a CA Enterprise Tester recording, trace or log. Because the volume serial number was obtained by successfully reading the JFCB associated with the data set, this problem should not occur. The error may indicate a problem with the VTOC of the volume containing the data set. Contact technical support personnel at your installation for assistance in resolving the problem.

PTE64020 - RECFM OF dsn IS NOT VALID FOR form

The record format (RECFM) of a data set, "dsn", specified as the target for a CA Enterprise Tester recording, trace or log, is not acceptable for the type of data being captured. When the message is issued, "form", is one of the values RECORD, TRACE, or LOG, indicating the type of data being captured. CA Enterprise Tester recordings may be written to RECFM=F or RECFM=FB data sets. CA Enterprise Tester traces may be written to RECFM=F, RECFM=FB, RECFM=FA or RECFM=FBA data sets. CA Enterprise Tester logs may be written to RECFM=FA or RECFM=FBA data sets. Standard blocking may also be specified for all capture data sets.

PTE64021 - LRECL OF dsn IS NOT VALID FOR form

The logical record length (LRECL) of a data set, "dsn", specified as the target for a CA Enterprise Tester recording, trace or log is not acceptable for the type of data being captured. When the message is issued, "form", is one of the values RECORD, TRACE, or LOG, indicating the type of data being captured. CA Enterprise Tester recordings must be written to data sets with LRECL=80 only. CA Enterprise Tester traces may be written to data sets with LRECL=133 through LRECL=256 (LRECL=135 is preferred). CA Enterprise Tester logs must be written to data sets with LRECL=151 only.

PTE64022 - BLKSIZE OF dsn IS NOT VALID

The block size (BLKSIZE) of a data set, "dsn", specified as the target for a CA Enterprise Tester recording, trace or log is not acceptable. For unblocked records, BLKSIZE must equal LRECL. For blocked records BLKSIZE must be a multiple of LRECL.

PTE64023 - dsn OPEN FAILED

An OPEN macro was issued for a data set, "dsn". At completion of the OPEN, the data set was not open. "dsn" is either a data set that was specified as the target for a CA Enterprise Tester recording, trace or log, or is a work data set if the target data set is a shared PDS. If the cause of the error can not be determined, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE64024 - function ABEND Sxxx ACCESSING dsn

A system abend, "Sxxx", occurred while accessing data set "dsn". "dsn" is either a data set that was specified as the target for a CA Enterprise Tester recording, trace or log, or is a work data set if the target data set is a member of a shared PDS. If "function" is OPEN, the abend occurred during data set open. If "function" is BLDL, the abend occurred while reading directory data for the specified target PDS member. Refer to IBM documentation of System Codes for an explanation of the system abend.

PTE64025 - I/O ERROR ON BLDL FOR dsn

An I/O error occurred during the execution of a BLDL issued to obtain current directory information for PDS member "dsn". "dsn" is a data set and member that was specified as the target for a CA Enterprise Tester recording, trace or log. It is probable that the directory of the PDS was previously corrupted and that data set repair is required before proceeding.

PTE64026 - WRITE ABEND Sxxx ACCESSING dsn

A system abend, "Sxxx", occurred while writing to data set "dsn". "dsn" was specified as the target for a CA Enterprise Tester recording, trace or log. The abend occurred while copying records from a work data set to the specified target data set. The work data set is retained. Captured data may be copied from the work data set to the target data set after resolving the problem with the target data set. Refer to IBM documentation of System Codes for an explanation of the system abend.

PTE64027 - WRITE ABEND Sxxx ACCESSING dsn

A system abend, "Sxxx", occurred while writing to data set "dsn". "dsn" is either the data set specified as the target for a CA Enterprise Tester recording, trace or log, or is a work data set if the specified target is a shared PDS. If the error occurred while writing to a work data set, the data set is retained and any captured data may be copied to the target data set. If a space related abend occurred while writing to a work data set, contact the systems programmer responsible for the installation and maintenance of CA Enterprise Tester at your installation so that work data set space parameters may be adjusted. If the abend occurred while writing to the specified target data set, and the target is a PDS member, an attempt was made retain all possible captured data by stowing the target member directory entry. Refer to IBM documentation of System Codes for an explanation of the system abend.

PTE64028 - WRITE ABEND Sxxx ACCESSING dsn

A system abend, "Sxxx", occurred while writing the last record or block of records to data set "dsn". "dsn" is either the data set specified as the target for a CA Enterprise Tester recording, trace or log, or is a work data set if the specified target is a shared PDS. If the error occurred while writing to a work data set, the data set is retained and any captured data may be copied to the target data set. If a space related abend occurred while writing to a work data set, contact the systems programmer responsible for the installation and maintenance of CA Enterprise Tester at your installation so that work data set space parameters may be adjusted. If the abend occurred while writing to the specified target data set, and the target is a PDS member, an attempt was made retain all possible captured data by stowing the target member directory entry. Refer to IBM documentation of System Codes for an explanation of the system abend.

PTE64029 - CLOSE ABEND Sxxx ACCESSING dsn

A system abend, "Sxxx", occurred while closing data set, "dsn". "dsn" is either the data set specified as the target for a CA Enterprise Tester recording, trace or log, or is a work data set if the specified target is a shared PDS. If the error occurred while writing to a work data set, the data set is retained and any captured data may be copied to the target data set. Refer to IBM documentation of System Codes for an explanation of the system abend.

PTE64030 - dsn IS BEING USED BY ANOTHER TASK

An attempt to obtain an exclusive data set ENQ on a PDS allocated with DISP=OLD, or an attempt to obtain a shared ENQ on a PDS allocated with DISP=SHR, was unsuccessful. The PDS, "dsn", was specified as the target for a CA Enterprise Tester recording, trace or log. The ENQ was attempted to protect the data set from inconsistent PDS share specifications when multiple recordings, traces or logs are concurrently active within a single address space. When a recording, trace or log is started without PDS sharing, no other recording, trace or log may specify the same PDS as a target until the first recording, trace or log is ended. When a recording, trace or log is started with PDS sharing, no other recording, trace or log may specify the same PDS as a target unless the request also specifies PDS sharing.

PTE64031 - dsn ALREADY EXISTS

A CA Enterprise Tester record, trace or log request specified that data be captured in PDS member, "dsn". The request also specified that an existing PDS member by the same name was to be protected. It was found that the PDS member already existed. The record, trace or log request is rejected and the user is informed of the existing target PDS member.

PTE64032 - dsn OPEN FAILED

An OPEN macro was issued for a data set, "dsn". At completion of the OPEN, the data set was not open. "dsn" was specified as the target for a CA Enterprise Tester recording, trace or log. If the cause of the error can not be determined, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE64033 - function ABEND Sxxx ACCESSING dsn

A system abend, "Sxxx", occurred while accessing data set "dsn". "dsn" was specified as the target for a CA Enterprise Tester recording, trace or log. Because the record, trace or log request specified protection for previously existing data, an attempt was made to check the directory of a target PDS for an existing member, or an attempt was made to read a sequential target data set to verify that no records existed. If "function" is OPEN, the abend occurred while opening the target data set. If "function" is BLDL, the abend occurred while reading directory data for the specified target PDS member. If "function" is READ, the abend occurred while reading the first record of a sequential target data set. In all cases, the record, trace or log request is aborted. Refer to IBM documentation of System Codes for an explanation of the system abend.

PTE64034 - I/O ERROR ON BLDL FOR dsn

An I/O error occurred during the execution of a BLDL issued to obtain current directory information for PDS member "dsn". "dsn" is a data set and member that was specified as the target for a CA Enterprise Tester recording, trace or log. It is probable that the directory of the PDS was previously corrupted and that data set repair is required before proceeding.

PTE64035 - dsn CONTAINS PROTECTED DATA

A CA Enterprise Tester record, trace or log request specified that data be captured in data set, "dsn". The request also specified that existing data in the data set was to be protected. It was found that the data set already contained data. The record, trace or log request is rejected and the user is informed of the existing data.

PTE64036 - CLOSE ABEND Sxxx ACCESSING dsn

A system abend, "Sxxx", occurred while accessing data set "dsn". "dsn" was specified as the target for a CA Enterprise Tester recording, trace or log. Because the record, trace or log request specified protection for previously existing data, an attempt was made to check the directory of a target PDS for an existing member, or an attempt was made to read a sequential target data set to verify that no records existed. After performing this processing, without error, the abend occurred while closing the data set. Refer to IBM documentation of System Codes for an explanation of the system abend.

PTE64037 - THE WORK DSN FROM PTEOM CAPTURE IS INVALID

A CA Enterprise Tester record, trace or log request specified that captured data was to be written to a member of a PDS and that the PDS was to be shared. A work data set is used when the target for a CA Enterprise Tester recording, trace or log is a shared PDS. The naming convention for work data sets is specified via the DSN parameter in the PTEOM CAPTURE statement in the CA Enterprise Tester Options Table installed at your installation. The data set name generated based on information contained in the PTEOM CAPTURE statement DSN parameter was an invalid data set name. The work data set name generated resulted in either no data set name or a data set name that exceeded 44 characters. Contact the systems programmer at your installation who is responsible for the installation and maintenance of CA Enterprise Tester, for assistance in resolving the problem.

PTE64038 - dsn IS BEING USED BY ANOTHER TASK

An attempt to obtain an exclusive ENQ on a CA Enterprise Tester work data set, "dsn", was not successful. The data set is in use by another CA Enterprise Tester task in the address space. A work data set is used when the target for a CA Enterprise Tester recording, trace or log is a shared PDS. Multiple work data sets are required when multiple recordings, traces or logs are concurrently written to shared PDSs. The naming convention for work data sets is specified via the DSN parameter in the PTEOM CAPTURE statement in the CA Enterprise Tester Options Table installed at your installation. The probable cause of this error is that the data set naming convention specified in the Options Table may generate the same work data set name for different record, trace or log requests. A work data set naming convention that uses the target member name as a node of the work data set name will likely produce unique work data set names for all active capture requests. Contact the systems programmer at your installation who is responsible for the installation and maintenance of CA Enterprise Tester, for assistance in resolving the problem.

PTE64039 - ALLOC FOR dsn FAILED, RC xxxxxxxx

An attempt to create a CA Enterprise Tester work data set, "dsn", was not successful. A work data set is used when the target for a CA Enterprise Tester recording, trace or log is a shared PDS. The naming convention for work data sets is specified via the DSN parameter in the PTEOM CAPTURE statement in the CA Enterprise Tester Options Table installed at your installation. The PTEOM CAPTURE statement also specifies work data set space and other parameters. A possible cause of this error may be the work data set parameters specified in the CA Enterprise Tester Options Table. When the message is issued, 'xxxxxxxx' is replaced by the SVC 99 error reason and information code. Refer to Appendix A of the help that is available when the HELP key is pressed while viewing the ISPF/PDF PRIMARY OPTION MENU or other IBM documentation of SVC 99 services for an explanation of the dynamic allocation error reason and information code. If the cause of the allocation error can not be determined, contact the systems programmer at your installation who is responsible for the installation and maintenance of CA Enterprise Tester, for assistance in resolving the problem.

PTE64040 - dsn DCB NOT VALID FOR WORK DS

A request to start a CA Enterprise Tester recording, trace or log specified that the captured data be written to a member of a shared PDS. When data is written to a shared PDS, CA Enterprise Tester uses a work data set to temporarily hold captured data. When the recording, trace or log is ended, the work data set is copied to the specified PDS member. While starting the recording, trace or log, the work data set, "dsn", was found to be an existing data set. However the RECFM, LRECL or BLKSIZE of the data set was not suitable for use as a work data set and was not a RECFM, LRECL or BLKSIZE used for any form of CA Enterprise Tester data capture. The data set is retained and the record, trace or log request is rejected. If the data set is a valid data set used for a purpose other than as a CA Enterprise Tester work data set, consult with the system programmer at your installation that is responsible for the installation and maintenance of CA Enterprise Tester, to determine if there is a conflict between data set names used by CA Enterprise Tester as work data sets and the data set named in this error message.

PTE64041 - UNABLE TO OBTAIN DSCB FOR dsn

An z/OS OBTAIN macro was issued to obtain the Format 1 DSCB from the VTOC of the volume containing data set, "dsn". The request failed. "dsn" is a work data set that was created to hold recorded, traced or logged data temporarily to allow a specified target PDS to be shared. It was found that the work data set already existed. The OBTAIN was issued to determine if the data set was suitable for reuse as a record, trace or log work data set. Because the volume serial number was obtained during successful dynamic allocation of the data set, this problem should not occur. The error may indicate a problem with the VTOC of the volume containing the data set. Contact technical support personnel at your installation for assistance in resolving the problem.

PTE64042 - dsn CONTAINS PROTECTED DATA

A CA Enterprise Tester record, trace or log request specified that data be captured in a member of a PDS and that the PDS was to be shared. When captured data is written to a shared PDS, data is initially written to a work data set. When the recording, trace or log is ended, the work data set is copied to the PDS member. When the current record, trace or log request was processed, and an attempt was made to create a work data set, it was found that the work data set, "dsn" already existed and that the existing work data set was not empty. The data contained in the existing work data set is probably previously captured data that could not be copied to a target PDS member because of a previous target PDS error. To avoid overlaying the previously captured data, the current record, trace or log request is rejected and the existing work data set is retained. If you wish to keep the data previously written to the work data set, copy the data to another data set. In any case, the work data set should be deleted before attempting to start another recording, trace or log that specifies the same target PDS member.

PTE64043 - dsn OPEN FAILED

An OPEN macro was issued for a data set, "dsn". At completion of the OPEN, the data set was not open. "dsn" is a work data set that was created to temporarily hold recorded, traced or logged data so that the specified target PDS could be shared. If the cause of the error can not be determined, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE64044 - function ABEND Sxxx ACCESSING dsn

A system abend, "Sxxx", occurred while accessing data set "dsn". "dsn" is a work data set that was created to hold recorded, traced or logged data temporarily to allow the specified target PDS to be shared. It was found that the work data set already existed. To protect any data in the existing work data set, an attempt was made to read the data set to verify that it was empty. The abend occurred while accessing the data set for this purpose. If "function" is OPEN, the abend occurred while opening the work data set. If "function" is READ, the abend occurred while reading the first record of the work data set. In any case, the work data set is retained and the record, trace or log request is aborted. Refer to IBM documentation of System Codes for an explanation of the system abend.

PTE64045 - CLOSE ABEND Sxxx ACCESSING dsn

A system abend, "Sxxx", occurred while accessing data set "dsn". "dsn" is a work data set that was created to hold recorded, traced or logged data temporarily to allow the specified target PDS to be shared. It was found that the work data set already existed. To protect any data in the existing work data set, an attempt was made to read the data set to verify that it was empty. After successfully verifying that the work data set was empty, the abend occurred while closing the work data set. The work data set is retained and the record, trace or log request is aborted. Refer to IBM documentation of System Codes for an explanation of the system abend.

PTE64046 - dsn OPEN FAILED

An OPEN macro was issued for a data set, "dsn". At completion of the OPEN, the data set was not open. "dsn" was specified as the target for a CA Enterprise Tester recording, trace or log. The error occurred while attempting to copy data captured in a work data set to a member of a shared PDS. The work data set is retained and the data can be copied to a desired target PDS when the cause of the error has been resolved. If the cause of the error can not be determined, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE64047 - function ABEND Sxxx ACCESSING dsn

A system abend, "Sxxx", occurred while accessing data set "dsn". "dsn" is a the data set and member that was specified as the target for a CA Enterprise Tester recording, trace or log. Because PDS sharing was specified, the captured data was written to a work data set. When the recording, trace or log was ended, an attempt was made to copy the data to the specified PDS member. The abend occurred during this processing. If "function" is OPEN, the abend occurred while opening the target data set. If "function" is BLDL, the abend occurred while reading directory data for the specified target PDS member. If "function" is WRITE, the abend occurred while writing records to the target data set. If "function" is CLOSE, the abend occurred while closing the target data set. In any case, the work data set is retained and the data can be copied to the target data set after the problem with the target data set has been corrected. Refer to IBM documentation of System Codes for an explanation of the system abend.

PTE64048 - I/O ERROR ON BLDL FOR dsn

An I/O error occurred during the execution of a BLDL issued to obtain current directory information for PDS member "dsn". "dsn" is a data set and member that was specified as the target for a CA Enterprise Tester recording, trace or log. Because sharing was specified for the PDS, recorded, traced or logged data was written to a work data set during capture. The error was detected when the recording, trace or log was ended and an attempt was made to copy the captured data to the specified target PDS member. It is probable that the directory of the PDS was previously corrupted and that data set repair is required before proceeding. The work data set is retained and captured data may be copied to the target data set after correcting the problem with the target PDS.

PTE64049 - dsn OPEN FAILED

An OPEN macro was issued for a data set, "dsn". At completion of the OPEN, the data set was not open. "dsn" is a work data set that was created to temporarily hold recorded, traced or logged data so that the specified target PDS could be shared. The error occurred while attempting to copy data captured in the work data set to a member of a shared PDS. The work data set is retained and the data can be copied to a desired target PDS if the cause of the error can be determined. If the cause of the error can not be determined, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE64050 - function ABEND Sxxx ACCESSING dsn

A system abend, "Sxxx", occurred while accessing data set "dsn". "dsn" is a work data set that was used to hold recorded, traced or logged data to allow sharing of the specified target PDS. When the recording, trace or log was ended, an attempt was made to copy the data to the specified PDS member. The abend occurred during this processing. If "function" is OPEN, the abend occurred while opening the work data set for input processing. If "function" is READ, the abend occurred while reading the work data set. In either case, the work data set is retained and the data can be possibly be copied to the target data set at a later time. Refer to IBM documentation of System Codes for an explanation of the system abend.

PTE64051 - dsn DIRECTORY IS FULL

An STOW macro instruction was issued to store a new or updated directory entry for PDS member, "dsn". "dsn" is a data set and member that was specified as the target for a CA Enterprise Tester recording, trace or log. It was found that the PDS directory was full and that the directory entry could not be stored. If PDS sharing was specified when the recording, trace or log was started, recorded, traced or logged data was written to a work data set during capture. When a work data set is used, the work data set is retained and the captured data may be copied to the target PDS after expanding the PDS directory.

PTE64052 - I/O ERROR ON STOW FOR dsn

An I/O error occurred during the execution of a STOW issued to store a new or updated directory entry for PDS member, "dsn". "dsn" is a data set and member that was specified as the target for a CA Enterprise Tester recording, trace or log. It is probable that the directory of the PDS was previously corrupted and that data set repair is required before proceeding. If PDS sharing was specified when the recording, trace or log was started, recorded, traced or logged data was written to a work data set during capture. When a work data set is used, the work data set is retained and the captured data may be copied to the target PDS after correcting the problem with the target PDS.

PTE64053 - STOW ABEND Sxxx ACCESSING dsn

A system abend, "Sxxx", occurred while accessing data set "dsn". "dsn" is the data set and member that was specified as the target for a CA Enterprise Tester recording, trace or log. After writing all data to the target PDS member, the abend occurred while attempting to stow the PDS member directory entry. The work data set is retained and the data can be copied to the target data set after the problem with the target data set has been corrected. Refer to IBM documentation of System Codes for an explanation of the system abend.

PTE64054 - KEEPING WORK DATA SET dsn

An I/O error occurred while recording, tracing or logging data to a PDS member. Because PDS sharing was specified, recorded, traced or logged data was written to a work data set, "dsn". When an I/O error occurs while capturing data and a work data set is being used, the work data set is retained. After correcting the data set error, described by another message issued at the time of the error, the work data set may be copied to a desired PDS member. This message is issued as a "write to programmer" message (WTO) to provide the name of the work data set to the user. The message is sent to a TSO user when their TSO PROFILE specifies WTPMSG. The message is suppressed when a user's TSO PROFILE specifies NOWTPMSG. After correcting a problem with a target PDS member and after disposing of data retained in the work data set, it is the user's responsibility to delete the work data set.

PTE64055 - I/O ERROR DELETING dsn(mem)

A request do delete token screen, 'dsn(mem)', using the STOW system service with the delete option, completed with RC=8, indicating that an I/O error was encountered during the delete operation. The I/O error implies probable corruption of the data set. Restore the dataset from a recent backup or delete and reallocate the data set before retrying the delete request.

PTE64100 - PTEOM PROFILE DSN IS INVALID

An attempt to generate a CA Enterprise Tester profile data set name as defined in the CA Enterprise Tester Options Table PTEOM PROFILE statement, failed because a valid data set name could not be created using the default or specified PROFILE DSN data. Correct the PTEOM PROFILE DSN data to define a valid CA Enterprise Tester PROFILE data set name.

PTE64101 - PROFILE DS, dsn, ALLOC FAILED, RC xxxxxxxx

An attempt to dynamically allocate a CA Enterprise Tester PROFILE data set "dsn", as defined in the CA Enterprise Tester Options Table PTEOM PROFILE statement, was unsuccessful. When the message is issued, 'xxxxxxxx' is replaced by the SVC 99 error reason and information code. Refer to Appendix A of the help that is available when the HELP key is pressed while viewing the ISPF/PDF PRIMARY OPTION MENU or other IBM documentation of SVC 99 services for an explanation of the dynamic allocation error reason and information code.

PTE64102 - PROFILE DCB, dsn, INVALID (NOT PS/FB/80)

An attempt to load a CA Enterprise Tester profile data set, 'dsn', failed because the dataset was not a sequential dataset or did not contain fixed length 80 character logical records (DSORG=PS, RECFM=F, LRECL=80), and is not valid for use as a CA Enterprise Tester PROFILE data set. CA Enterprise Tester profile data sets are created automatically for each user when CA Enterprise Tester is first used, and are maintained and updated automatically by CA Enterprise Tester. The name of a Profile data set is specified at an installation via the CA Enterprise Tester Options Table PTEOM PROFILE statement. If a data set other then a CA Enterprise Tester Profile dataset has been named to conflict with the CA Enterprise Tester Options Table PTEOM PROFILE DSN data, the other dataset must renamed in order to use CA Enterprise Tester. If the data set is a CA Enterprise Tester profile data set that has been corrupted, you may build a new profile dataset by first deleting the invalid dataset, then by assigning token data to each(any) token screen contained in the CA Enterprise Tester token dataset displayed at the CA Enterprise Tester Token Security panel.

PTE64103 - PROFILE DS, dsn, IS UNREADABLE

An I/O error was encountered while attempting to load a CA Enterprise Tester profile data set, 'dsn', and is therefore unreadable. CA Enterprise Tester profile data sets are created automatically for each user when CA Enterprise Tester is first used, and are maintained and updated automatically by CA Enterprise Tester. The name of a Profile data set is specified at an installation via the CA Enterprise Tester Options Table PTEOM PROFILE statement. If a data set other then a CA Enterprise Tester Profile dataset has been named to conflict with the CA Enterprise Tester Options Table PTEOM PROFILE DSN data, the other dataset must renamed in order to use CA Enterprise Tester. If the data set is a CA Enterprise Tester profile data set that has been corrupted, you may build a new profile dataset by first deleting the invalid dataset, then by assigning token data to each(any) token screen contained in the CA Enterprise Tester token dataset displayed at the CA Enterprise Tester Token Security panel.

PTE64104 - PROFILE STORE TO DS dsn(mem) FAILED (I/O ERROR)

An I/O was encountered while attempting to store a CA Enterprise Tester profile data set, 'dsn'. As a result, profile data has probably been lost. CA Enterprise Tester profile data sets are created automatically for each user when CA Enterprise Tester is first used, and are maintained and updated automatically by CA Enterprise Tester. The name of a Profile data set is specified at an installation via the CA Enterprise Tester Options Table PTEOM PROFILE statement. If the data set is a CA Enterprise Tester profile data set that has been corrupted, you may build a new profile dataset by first deleting the invalid dataset, then by assigning token data to each(any) token screen contained in the CA Enterprise Tester token dataset displayed at the CA Enterprise Tester Token Security panel.

PTE64105 - PTEOM dataid DSN IS INVALID

An attempt to create a new CA Enterprise Tester dataset failed because a valid data set name could not be generated using default or specified Options Table DSN data. When the message is issued, 'dataid', is replaced by PROFILE, TOKEN, TRACE, RECORD or LOG, identifying the PTEOM Options Table statement whose DSN data does not resolve to a valid data set name. Correct the PTEOM 'dataid' DSN data to define a valid data set name.

PTE64106 - ALLOC FOR NEW dataid DS dsn FAILED, RC xxxxxxxx

An attempt to create a new CA Enterprise Tester dataset using dynamic allocation failed. When the message is issued, 'dataid', is replaced by PROFILE, TOKEN, TRACE, RECORD or LOG, identifying the PTEOM Options Table statement that defined the data set to be created, 'dsn', and 'xxxxxxxx' is replaced by the SVC 99 error reason and information code. Refer to Appendix A of the help that is available when the HELP key is pressed while viewing the ISPF/PDF PRIMARY OPTION MENU or other IBM documentation of SVC 99 services for an explanation of the dynamic allocation error reason and information code.

PTE64107 - NEW dataid DS, dsn(mem), OPEN/CLOSE FAILED

After creating a new CA Enterprise Tester profile or other dataset, 'dsn', that was defined as sequential, an attempt to initialize the data set by opening and closing it failed. When the message is issued, 'dataid', is replaced by PROFILE, TOKEN, TRACE, RECORD or LOG, identifying the PTEOM Options Table statement that defined the data set being created. Determine the cause of the abend and correct the CA Enterprise Tester Options Table statement if required, or delete the new dataset and retry the request when the cause of the I/O error has been resolved. You may optionally create the data set manually using ISPF data set utilities.

PTE65001 dsn NOT FOUND

An attempt to resolve the relative generation number of a CA Enterprise Tester trace data set or trace list, by locating the catalog entry for the data set name specified, failed. When the message is issued, 'dsn' is replaced by the data set name specified. Either the data set is not cataloged or the relative generation number expression specified was invalid. A member of a partitioned data set should be specified via a separate field and should not be included in parenthesis immediately following the data set name.

PTE65002 - RC eeeeiiii ALLOCATING dsn

An attempt to allocate a CA Enterprise Tester trace data set or trace list failed. When the message is issued, 'eeeeiiii' is replaced by the SVC 99 error reason and information code, and 'dsn' is replaced by the specified data set name. Refer to Appendix A of the help that is available when the HELP key is pressed while viewing the ISPF/PDF PRIMARY OPTION MENU or other IBM documentation of SVC 99 services for an explanation of the error.

PTE65003 - RC eeeeiiii ALLOCATING dsn

An attempt to allocate a CA Enterprise Tester trace data, whose name was specified via an ANALYZE statement in a CA Enterprise Tester trace list, failed. When the message is issued, 'eeeeiiii' is replaced by the SVC 99 error reason and information code, and 'dsn' is replaced by the specified data set name. Refer to Appendix A of the help that is available when the HELP key is pressed while viewing the ISPF/PDF PRIMARY OPTION MENU or other IBM documentation of SVC 99 services for an explanation of the error.

PTE65004 - ABEND Sxxx OPENING dsn

An abend occurred while opening a CA Enterprise Tester trace data set or trace list. When the message is issued, 'Sxxx' is replaced by the system completion code and 'dsn' is replaced by the name of the data set being opened. One common system completion code, S013, most likely means that a PDS member was specified, however the specified member does not exist. Refer to IBM documentation of System Codes for an explanation of other system completion codes.

PTE65005 - dsn OPEN FAILED

An attempt to open a CA Enterprise Tester trace data set or trace list failed. When the message is issued, 'dsn' is replaced by the name of the data set being opened. When the open was issued, an open abend did not occur, however after issuing the open request, it was found that the data set was not open. If the reason for the open failure can not be determined, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE65006 - dsn IS NOT A TRACE OR LIST

A specified data set was not a CA Enterprise Tester trace data set or trace list. Either the data set was not a sequential data set or PDS member, the data set did not contain fixed length records or the data set LRECL was not valid. Trace list data set records must be from 80 to 256 bytes long. Trace data set records must be from 135 to 256 bytes long. A data set is considered a trace list and not a trace data set if the first character of the first record in the data set or PDS member begins with an asterisk or an alphabetic character. When the message is issued, 'dsn' is replaced by the name of the data set specified.

PTE65007 - dsn TRACE LIST INVALID

A specified data set was recognized as a trace list and not a trace data set because the first character of the first record in the data set was an asterisk or an alphabetic character. However, another record in the data set did not begin with an asterisk or an alphabetic character, or a trace list ANALYZE statement specified an invalid data set name or partitioned data set member name. When the message is issued, 'dsn' is replaced by the name of the invalid trace list that was being processed when the error was detected.

PTE65008 - dsn NOT A PDS

A request specified a PDS member name, however the data set specified was not a Partitioned Data Set. When the message is issued, 'dsn' is replaced by the data set and member name specified. Specify a member name only when the specified data set is a PDS.

PTE65009 - SPECIFY MEMBER IN dsn

A request specified a trace or trace list data set that was a partitioned data set. However, the request did not specify a member name. When the message is issued, 'dsn' is replaced by the data set name specified. When a PDS is specified, a member name must also be specified.

PTE65010 - dsn NOT A PDS

A request specified a PDS member name, however the data set specified was not a Partitioned Data Set. When the message is issued, 'dsn' is replaced by the data set and member name specified. Specify a member name only when the specified data set is a PDS.

PTE65011 - SPECIFY MEMBER IN dsn

A request specified a trace or trace list data set that was a partitioned data set. However, the request did not specify a member name. When the message is issued, 'dsn' is replaced by the data set name specified. When a PDS is specified, a member name must also be specified.

PTE65012 - dsn IS NOT A TRACE

A request specified a data set that should be either a CA Enterprise Tester trace data set or a trace list. However, the data set is neither a sequential or partitioned data set and therefore can not be a trace data set or a trace list. When the message is issued, 'dsn' is replaced by the specified data set name.

PTE65013 - ABEND Sxxx OPENING dsn

An abend occurred while opening a CA Enterprise Tester trace data set. When the message is issued, 'Sxxx' is replaced by the system completion code and 'dsn' is replaced by the name of the data set being opened. One common system completion code, S013, most likely means that a PDS member was specified, however the specified member does not exist. Refer to IBM documentation of System Codes for an explanation of other system completion codes.

PTE65014 - dsn OPEN FAILED

An attempt to open a CA Enterprise Tester trace data set. When the message is issued, 'dsn' is replaced by the name of the data set being opened. When the open was issued, an open abend did not occur, however after issuing the open request, it was found that the data set was not open. If the reason for the open failure can not be determined, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE65015 - dsn NOT A TRACE DATASET

A request specified a data set that must be a CA Enterprise Tester trace data set. However it was found that the data set was not valid. When the message is issued, 'dsn' was replaced by the name of the data set being processed when the error was detected. The data set specified must be a CA Enterprise Tester trace data set that was produced using the CA Enterprise Tester FORMAT trace option. FORMAT is the default option specified when the enter key is pressed at the CA Enterprise Tester Interactive System Trace Panel. The trace data set must be sequential data set or PDS member with LRECL 135 through 256. IMAGE and UPPERCASE traces are not acceptable input to the process requested because they do not contain the screen attributes and other control data required to perform the function that was requested.

PTE65016 - dsn ALTERED WHILE READING

While browsing a CA Enterprise Tester trace data set or trace list, it was found that a trace data set was altered while the browse was in progress. When the message was is issued, 'dsn' is replaced by the data set being processed when the data set alteration was detected. When a data set is altered during a browse, it becomes impossible to keep track of the relative screen number of screens in the trace data set(s) being processed. The browse is truncated to include only screens that precede the detected alteration. The browse may be ended and restarted to refresh CA Enterprise Tester controls that support random access to the trace data set(s).

PTE65017 - dsn TRACE DATA INVALID

A specified CA Enterprise Tester trace data set contained invalid data. When the message is issued, 'dsn' is replaced by the data set being processed when the error was detected. The input to the function requested must be a CA Enterprise Tester trace data set that was produced using the CA Enterprise Tester FORMAT trace option. FORMAT is the default option specified when the enter key is pressed at the CA Enterprise Tester Interactive System Trace Panel. The trace data set must be sequential data set or PDS member with LRECL 135 through 256. IMAGE and UPPERCASE traces are not acceptable input to the function requested because they do not contain the screen attributes and other control data required to perform the function that was requested.

PTE65018 - ABEND Sxxx READING dsn

An abend occurred while reading a CA Enterprise Tester trace data set. When the message is issued, 'Sxxx' is replaced by the system completion code and 'dsn' is replaced by the name of the data set being read. Refer to IBM documentation of System Codes for an explanation of the system completion codes shown in the message.

PTE65019 - pack-error IN dsn

A CA Enterprise Tester trace data set was being processed that was written using the ISPF packed data format. However, en error was detected while unpacking the data. When the message is issued, 'pack-error' is replaced by text that identifies the packed data token that was invalid, and 'dsn' is replaced by the name of the data set that was being processed when the error was detected. A data set may have been 'truncated' if the system went down while capturing data or if an abend (ie., SD37) occurred while writing to the trace data set. When the error is detected, access to subsequent records in the data set or to subsequent data sets in a trace list is avoided. If the cause of the error can not be determined and the error persists after recreating the invalid trace data set, contact CA Enterprise Tester technical support for assistance in resolving the problem.

PTE65020 - dsn IS NOT DSORG=PO,RECFM=FB,LRECL=135

A data set was specified via the CA Enterprise Tester Token Security Panel, that was not a valid token screen dataset. Either the dataset was not a PDS or PDSE, was not RECFM=F or was not LRECL=135-256. To assign a value to a CA Enterprise Tester security token, to change the value of a token, or to delete a token, specify a valid dataset that contains a traced token screen image.

PTE65021 - dsn(mem) WAS NOT FOUND

A valid partitioned token screen dataset was specified via the CA Enterprise Tester Token Security Panel, however the member name specified was not found in the dataset. Either specify the name of an existing token screen member or leave the member name field blank or specify a member name pattern to display a list of existing dataset members.

PTE65022 - dsn IS AN INVALID DATASET NAME

The data set name specified at the CA Enterprise Tester Token Security Panel is an invalid data set name. To proceed, specify the name of a valid token screen data set containing traced token screen image(s) to be viewed, changed or deleted.

PTE65023 - TRACE LINE condition

While attempting to browse a CA Enterprise Tester trace data set or display a CA Enterprise Tester token screen, an invalid trace line was found. When the message is issued, 'condition', is replaced by text that states the specific reason that the trace line is considered to be invalid. Either correct or delete the invalid trace line from the trace data set or member, or if the data set is not a CA Enterprise Tester trace or token data set, specify a valid CA Enterprise Tester trace or token data set name and member.