Workload Simulator

Utilities Guide

Version 1 Release 1

SC31-8947-01

Workload Simulator

Utilities Guide

Version 1 Release 1

SC31-8947-01

Note!
Before using this information and the product it supports, be sure to read the general information under "Notices" on page 285.

Second Edition (October 2015)

This document applies to the Workload Simulator Version 1 Release 1 (program number 5655-I39), an IBM licensed program, which runs under the following operating systems:

MVS/370 (MVS/SP Version 1 or later)

MVS/Extended Architecture (MVS/SP Version 2 or later)

MVS/Enterprise System Architecture (MVS/SP Version 3 or later)

OS/390

Publications are not stocked at the address given below. If you want more IBM publications, ask your IBM representative or write to the IBM branch office serving your locality.

Contents

Figures

Tables

About this book

This book describes the Workload Simulator (WSim) general utility programs.

This book is intended to help customers use the WSim general utilities and provide an overview of the utilities that assist in generating message generation decks and STL procedures. The WSim utilities covered are the WSim/ISPF Interface, the Preprocessor and ITPSYSIN, the Loglist Utility, the Log Compare Utility, the Response Time Utility and ITPECHO. It contains information about how to run the utilities, including the control commands to achieve the desired results.

This book also helps customers use the WSim script generating utilities to produce WSim message generation decks and STL procedures from captured data. The utilities covered are the Interactive Data Capture Utility, the Log Script Generator Utility, the SNA 3270 Reformatter Utility and the Script Generator Utility. It contains information on how to run the utilities, including control commands to achieve the desired results.

Who should read this book

This book is intended for operators who use the WSim general utilities, for programmers who write routines for the WSim general utility programs and for users who will be capturing data and using the captured data to produce message generation decks or STL procedures. The book assumes that you are familiar with WSim and with the statements and commands used in message generation decks and STL procedures.

For more information about networks and message generation decks, see *Creating WSim Scripts* and the *WSim Script Guide and Reference*.

How to use this book

The purpose of part 1 of this book is to introduce the WSim general utilities, explain how to use the utilities, and describe the output produced by the utilities.

This part contains the following chapters:

- Chapter 1, “Introducing WSim utilities,” on page 3 provides an overview of the WSim general utility programs.
- Chapter 2, “Running WSim with the WSim/ISPF Interface,” on page 5 provides an overview of the WSim/ISPF Interface.
- Chapter 3, “Using the Preprocessor and ITPSYSIN to preprocess WSim scripts,” on page 11 describes the Preprocessor. The Preprocessor checks the syntax of your network definition statements and message generation decks and stores them in the appropriate data sets. This part also describes how to use ITPSYSIN, a subset of the Preprocessor. This program stores network definition statements and message generation decks in the appropriate data sets without checking syntax.
- Chapter 4, “Using the Loglist Utility to format the log data set,” on page 25 and Chapter 5, “Specifying loglist control commands,” on page 45 describes the Loglist Utility, which produces a formatted report of the activity recorded on the log data set.

- Chapter 6, "Using the Log Compare Utility to compare log data sets," on page 63 and Chapter 7, "Specifying Log Compare Utility control commands," on page 91 describes the Log Compare Utility, which compares the display records in two log data sets and reports the differences.
- Chapter 8, "Using the Response Time Utility to analyze response times," on page 107 and Chapter 9, "Specifying Response Time Utility control commands," on page 129 describes the Response Time Utility, a program that produces a response time analysis report based on the activity recorded in the WSim log data set.
- Chapter 10, "Using ITPECHO to test WSim simulated resources," on page 151 describes ITPECHO, a VTAM® application program supplied with WSim as a sample routine to help you understand WSim and its installation and planning processes.
- Chapter 11, "Simulated resource type codes," on page 161 lists the resource codes used in WSim.
- Chapter 12, "Understanding message logging," on page 163 provides background information related to message logging. This information includes:
  – What message logging is
  – How messages are time stamped
  – How messages are written to the log data set, including messages for special devices.
- | Chapter 13, "Using the TCP/IP Trace Utility," on page 167 provides an overview | of the TCP/IP Trace Utility. The TCP/IP Trace Utility capture TCP/IP data trace | records by using TCP/IP trace Network Management Interface(NMI).
- | Chapter 14, "Using the TCP/IP Trace Formatting Utility," on page 171 provides an overview of TCP/IP Trace Formatting Utility. The TCP/IP Trace Formatting Utility produces a formatted report of the TCP/IP trace records that are saved in | a data set.

The purpose of part 2 of this book is to help customers use the WSim script generating utilities to produce WSim message generation decks and STL procedures from captured data.

This part contains the following chapters:

- Chapter 15, "Generating scripts interactively with IDC," on page 175 provides an overview of the Interactive Data Capture Utility. The Interactive Data Capture Utility captures 3270 SNA session data between a terminal and a host application and generates a WSim message generation deck or an STL program from the captured data.
- Chapter 16, "Generating scripts from IDC or WSim log data sets," on page 213 provides an overview of the Log Script Generator Utility. The Log Script Generator Utility generates a WSim message generation deck or an STL program from an IDC or WSim log data set.
- Chapter 17, "Generating 3270 scripts from captured traces," on page 221 provides an overview of the SNA 3270 Reformatter Utility. The SNA 3270 Reformatter Utility creates a WSim log data set from captured log data, which can be used by the Log Script Generator Utility to generate a WSim message generation deck or an STL program.
- Chapter 18, "Using the Script Generator Utility," on page 231 provides an overview of the Script Generator Utility. The Script Generator Utility helps you create message generation decks by using captured data traffic and network configuration data. You must follow five steps to use the Script Generator Utility:

1. Obtain a trace of system activity.
2. Reformat the captured data.
3. Sort the reformatted data.
4. Define the network for your simulation.
5. Generate the message generation decks.

A utility program (ITPVTBRF) is provided as part of the Script Generator Utility to help perform the steps in this procedure.

- Chapter 19, "3270 password masking," on page 271 provides an overview of the Password Masking. Password Masking helps mask potential passwords entered by users of the WSim data capture and script generation utilities and mask their presence in generated WSim scripts, simulation data views, and output reports.
- Chapter 20, "Work Station Trace Reformatter Utility," on page 275 provides an overview of Work Station Trace Reformatter Utility (ITPWSTRF) reformats OS/2 Communications Manager and IBM Communications Server LU 6.2 traces into TIR formatted records for processing by using the WSim Script Generator.
- Chapter 21, "Generating STL from TCP/IP traces," on page 277 describes how to generate STL programs from a TCP/IP trace by using the TCP/IP Trace STL Generation Utility (ITPIPGEN) and how to use control commands.

Where to find more information

The following list shows the books in the WSim library. For more information about related publications, see the "Bibliography" on page 295.

Planning, Installation, and Operation	
WSim User's Guide	SC31-8948
WSim Test Manager User's Guide and Reference	SC31-8949
WSim Messages and Codes	SC31-8951
Resource and Message Traffic Definition	
Creating WSim Scripts	SC31-8945
WSim Script Guide and Reference	SC31-8946
WSim Utilities Guide	SC31-8947
Customization	
WSim User Exits	SC31-8950

Part 1. General utilities

Chapter 1. Introducing WSim utilities

This chapter introduces the programs that make up the WSim utilities.

What is Workload Simulator?

Workload Simulator (WSim) is a terminal simulation tool. You can use WSim to determine system performance and response time, to perform functional testing, and to automate regression testing. Used as a basic tool in a comprehensive test plan, WSim increases the effectiveness of system testing by providing a structured and systematic approach to all phases of system testing.

WSim Version 1 Release 1 runs on any IBM® host processor that supports:

- MVS/370 (MVS/SP Version 1 or later)
- MVS/XA (MVS/SP Version 2 or later)
- MVS/ESA (MVS/SP Version 3 or later)
- OS/390

In this book, MVS™ is any environment running MVS/370 or MVS/XA (unless explicitly stated otherwise).

WSim general utilities programs

The WSim general utilities described in this book consist of the following:

WSim/ISPF Interface
The WSim/ISPF Interface provides a CUA-compliant operator interface to many functions of WSim.

Preprocessor
The Preprocessor checks the syntax of your network definition statements and message generation decks and stores them in the appropriate data sets.

ITPSYSIN
ITPSYSIN contains a subset of the preprocessor functions. This program stores the network definition statements and message generation decks in the appropriate data sets without checking the syntax.

Loglist Utility
The Loglist Utility produces formatted reports of the activity recorded in the WSim log data set. You can request a report for all the resources in the WSim log data set or for only specific network resources.

Log Compare Utility
The Log Compare Utility compares the 3270 display records in two log data sets and reports the differences. You can specify the records and the fields that the Log Compare Utility compares, using user-defined parameters.

Response Time Utility
The Response Time Utility produces a response time analysis report based on the activity recorded in the WSim log data set.

ITPECHO
ITPECHO is a VTAM application that helps you understand WSim and its installation and planning processes.

TCP/IP Trace Utility
The TCP/IP Trace Utility captures TCP/IP data trace records for the messages that are exchanged between a client and a server.

TCP/IP Trace Formatting Utility
The TCP/IP Trace Formatting Utility formats TCP/IP trace records that are saved to a data set.

Script generating utility programs

This section describes the utility programs that you can use to create message generation decks: Interactive Data Capture, the Log Script Generator Utility and the Script Generator Utility. Interactive Data Capture and the Log Script Generator Utility can also generate STL procedures. The SNA 3270 Reformatter Utility is also described.

Interactive Data Capture

The Interactive Data Capture Utility captures 3270 SNA session data between a terminal and a host application and generates a WSim message generation deck or an STL program from the captured data.

Log Script Generator Utility

The Log Script Generator Utility generates a WSim message generation deck or an STL program from an IDC or WSim log data set.

SNA 3270 Reformatter Utility

The SNA 3270 Reformatter Utility creates a WSim log data set from captured log data, which can be used by the Log Script Generator Utility to generate a WSim message generation deck or an STL program.

Script Generator Utility

The Script Generator Utility helps you create message generation decks by using captured data traffic and network configuration data. You must follow five steps to use the Script Generator Utility:

1. Obtain a trace of system activity.
2. Reformat the captured data.
3. Sort the reformatted data.
4. Define the network for your simulation.
5. Generate the message generation decks.

A utility program (ITPVTBRF) is provided as part of the Script Generator Utility to help perform the steps in this procedure.

TCP/IP Trace Script Generator Utility

The TCP/IP TraceScript Generator Utility reads TCP/IP trace records from a data set and generates an STL program that replicates the transactions between a client and server.

Chapter 2. Running WSim with the WSim/ISPF Interface

The WSim/ISPF Interface is a CUA-compliant operator interface that lets you start the WSim utilities. To use the WSim/ISPF Interface, you need to run WSim under MVS with TSO/E and ISPF. Refer to *WSim User's Guide* for information about setting up your environment to run the WSim/ISPF Interface.

You can access the following WSim functions with the WSim/ISPF Interface:

1. STL Translator
2. Preprocessor and ITPSYSIN
3. Interactive Data Capture (IDC) Utility
4. Log Script Generator Utility
5. Script Generator Utility
6. SNA 3270 Reformatter Utility
7. WSim simulation runs
8. Response Time Utility
9. Log Compare Utility
10. Loglist Utility

You can get help for the WSim/ISPF Interface at any time by pressing F1. For more information, see "Getting help from the WSim/ISPF Interface" on page 7.

Invoking the WSim/ISPF Interface

Invoke the WSim/ISPF Interface by entering the following command on the TSO command line:

```
ITP0MAIN prefix (PROFILE DEBUG
```

prefix is the installation qualifier for the WSim/ISPF Interface defaults data set. If you do not specify *prefix*, "WSIM$$$$" is used. (You might invoke the WSim/ISPF Interface differently at your installation. See your system programmer for this information or refer to *WSim User's Guide* for instructions on setting up the WSim/ISPF Interface.)

Specify PROFILE to read the WSim/ISPF Interface defaults data set (created by the ITP0INST exec) and use the values within that data set as defaults. If you do not specify PROFILE, the defaults data set is only read the first time you invoke the WSim/ISPF Interface for the current release.

You may make changes to the WSim/ISPF Interface source code. However, IBM is under no obligation to support the WSim/ISPF Interface code you change. If you need to debug a problem encountered in your modified code, specify DEBUG to help you find the problem.

You can specify DEBUG and PROFILE in either order.

The first time you invoke the WSim/ISPF Interface, you see a logo panel. Press Enter to display the WSim/ISPF Interface main panel, as shown in Figure 1 on page 6. You can select the following options from this panel:

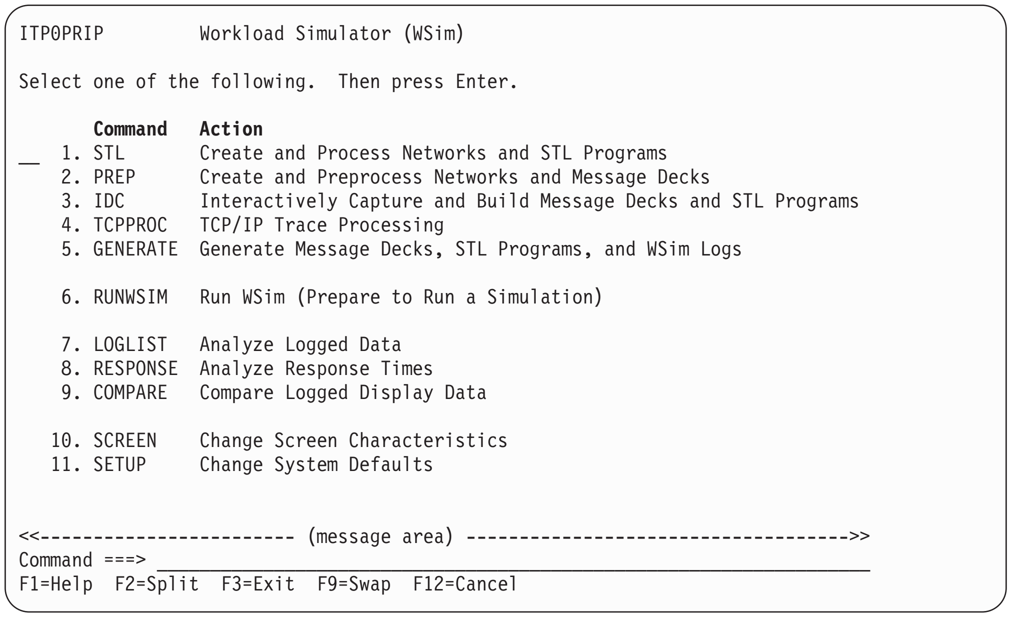

```
ITP0PRIP        Workload Simulator (WSim)

Select one of the following.  Then press Enter.

       Command   Action
__  1. STL       Create and Process Networks and STL Programs
    2. PREP      Create and Preprocess Networks and Message Decks
    3. IDC       Interactively Capture and Build Message Decks and STL Programs
    4. TCPPROC   TCP/IP Trace Processing
    5. GENERATE  Generate Message Decks, STL Programs, and WSim Logs

    6. RUNWSIM   Run WSim (Prepare to Run a Simulation)

    7. LOGLIST   Analyze Logged Data
    8. RESPONSE  Analyze Response Times
    9. COMPARE   Compare Logged Display Data

   10. SCREEN    Change Screen Characteristics
   11. SETUP     Change System Defaults


<<------------------------ (message area) ------------------------------------>>
Command ===> __________________________________________________________________
F1=Help  F2=Split  F3=Exit  F9=Swap  F12=Cancel
```

Figure 1. WSim/ISPF Interface main panel

STL	This option lets you create and translate network definitions and STL programs, using the STL Translator. For more information on this, refer to *WSim Script Guide and Reference*.
PREP	This option lets you create and translate network definitions and WSim message generation decks, using the Preprocessor or ITPSYSIN. For more information on this, see Chapter 3, "Using the Preprocessor and ITPSYSIN to preprocess WSim scripts," on page 11.
IDC	This option lets you generate message generation decks or STL programs interactively, using the Interactive Data Capture Utility. For more information on this, refer to Part 2, "Script generating utilities," on page 173.
TCPPROC	This option lets you create and format TCP/IP traces. For more information refer to Part 1, "General utilities," on page 1.
GENERATE	This option lets you generate message generation decks or STL programs, using one of the WSim script generation utilities. For more information on this, refer to Part 2, "Script generating utilities," on page 173.
RUNWSIM	This option lets you start a WSim simulation run. For more information on this, refer to *WSim User's Guide*.
LOGLIST	This option lets you format a WSim log data set, using the Loglist Utility. For more information on this, see Chapter 4, "Using the Loglist Utility to format the log data set," on page 25.
RESPONSE	This option lets you calculate response times, using the Response Time Utility. For more information on this, see Chapter 8, "Using the Response Time Utility to analyze response times," on page 107.
COMPARE	This option lets you compare log data sets, using the Log Compare Utility. For more information on this, see Chapter 6, "Using the Log Compare Utility to compare log data sets," on page 63.
SCREEN	This option lets you change such screen characteristics as color, function key display, panel ID display, and beep status.
SETUP	This option lets you set up system default values, such as data set names, installation qualifier and printer information.

Getting help from the WSim/ISPF Interface

If you are currently on an application panel (for example, the STL panel) and you press the F1 function key, you see a help panel. The type of help you get depends on the location of the cursor:

Cursor Location	Type of Help
Input field	Help on that input field.
Message line	Help on the message, if one is displayed. If not, help on the application panel.
Command line	Help on the available commands.
Anywhere else	Help on the application panel.

Once you invoke Help, you may choose from several different types of help. If you press F1 while inside Help, you see a Help for Help panel, which gives you further information on how to access the various help items.

Navigating the panels

Figure 2 shows the panel structure for the WSim/ISPF Interface application panels.

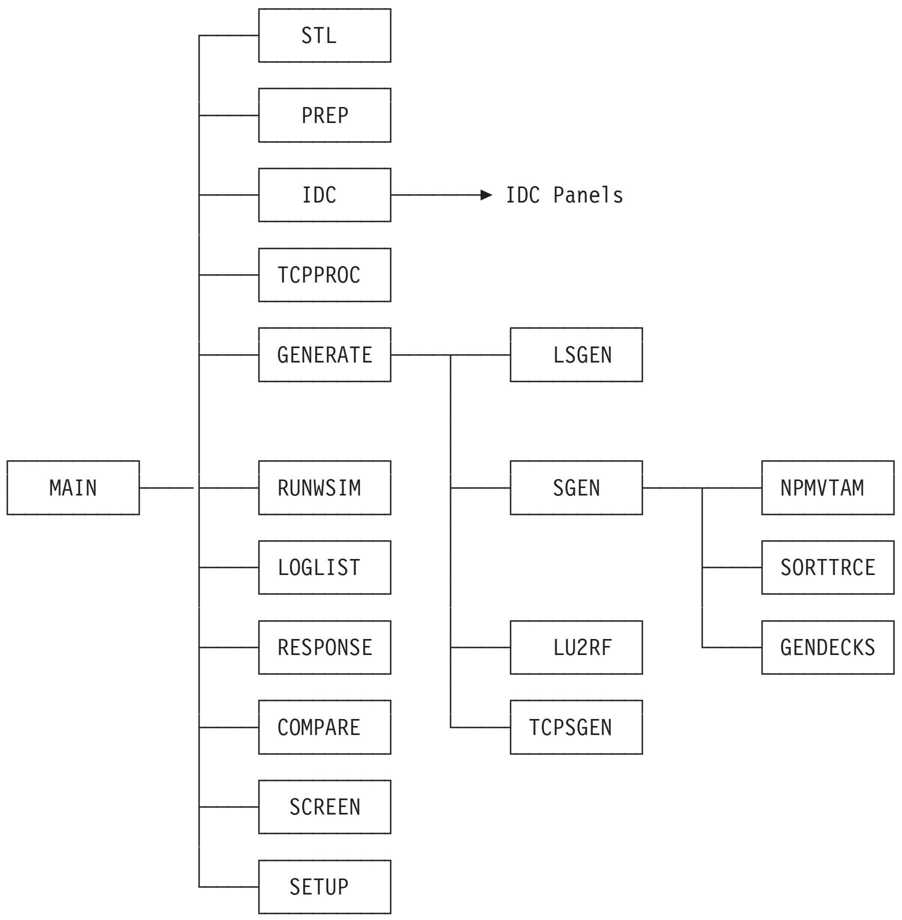


Figure 2. Structure of WSim/ISPF Interface panels

Note: There is no hierarchical structure for the Help panels.

You may navigate among the panels by using the function keys (F3 or F12), by entering a selection on the MAIN, GENERATE, or SGEN panel, or by entering a command on the command line and pressing Enter.

There are more panels that are not shown in Figure 2 on page 7:
- Models
- Specify Additional Networks and STL Programs (STL)
- Specify Additional Message Decks (RUNWSIM)
- Specify Additional Networks (RUNWSIM)
- Specify Additional Network and Message Decks (PREP)
- Control Analysis of Logged Data
- Specify Additional Loglist Groups
- Submit a Batch Job.

These panels are all full-screen windows. Selecting F4 (Edit Input) with the Model New Input field set to Y while on the STL or PREP panel with a valid data set name and new member specified on the Input Data Set field causes the Models window to appear. On this window, you can select a model to use to edit your input. If you position your cursor on an Input Data Sets field and press F10 on the STL, PREP, or RUNWSIM panels, a window appears allowing foradditional Input Data Sets to be specified. On this window, you can enter additional data set names for that field. Input fields with additional input data sets defined are marked with a "+" to the left of the input field description.

If you enter a P in the Control input area field and a Y in the Display control panel fields of the LOGLIST panel, you see the Control Analysis of Logged Data window, allowing you to specify options to use when you run the Loglist Utility. If you press F10 on this window, you see the Specify Additional Loglist Groups window, allowing you to specify additional devices, LUs, or TPs to be formatted for the Loglist Utility. For all utilities, if you select to run in batch mode and press Enter, you see a window, allowing you to change information before you submit the batch job.

Entering data on panel fields

Some input fields are always optional, some are optional depending on input to another field, and some are always required. You can distinguish required fields from optional fields by viewing help information for that field. Default values are provided for fields requiring input.

Entering and processing commands on the command line

Whenever you type a command on the command line and press Enter, the command is processed and everything else on the panel is ignored. If you press a function key instead of Enter after you type a command on the command line, that command is ignored and the action specified by the function key is performed.

Entering data set information

To fully qualify data set entry fields, you must enclose the data set name in single quotes. If the data set name is unqualified (that is, not enclosed in quotes), the TSO user prefix is attached to the name unless you specify NOPREFIX in your user profile. The maximum length for data set names is 44 characters, including the TSO user prefix but not including member names.

Using function keys

You can use the following function keys in the WSim/ISPF Interface:

F1 (Help)	Invokes help information on the WSim/ISPF Interface. See "Getting help from the WSim/ISPF Interface" on page 7 for more information about getting help.
F2 (Split)	Splits the screen at the cursor position, except for help panels and full-screen windows.
F3 (End)	Ends the WSim/ISPF Interface from any application panel that it is displayed on. Pressing F3 while you are on a help panel ends the help session and returns you to the panel you were on when you invoked help.
F4 (Edit Input)	Edits the input data set indicated by the cursor position, using the ISPF/PDF editor with your current edit profile. If you edit a partitioned data set without specifying a member name, you see a member selection list displayed on your screen.
F5 (Restore)	Restores previously saved field values on a panel.
F6 (Browse output)	Browses printer output or models, using the ISPF/PDF browser. If you browse a partitioned data set without specifying a member name, you see a member selection list displayed on your screen.
F7 (Backward)	Scrolls a panel backward. An indicator, "More: –", tells you if you can scroll backward on a panel.
F8 (Forward)	Scrolls a panel forward. An indicator, "More: +", tells you if you can scroll forward on a panel.
F9 (Swap)	Swaps two previously split panels.
F10 (Edit Control File)	Edits a control file, using the ISPF/PDF editor with your current edit profile. If you edit a partitioned data set without specifying a member name, you see a member selection list displayed on your screen.
F10 (Additional Input)	Allows input of additional data sets for the field indicated by the cursor position (see "Navigating the panels" on page 7 for more information).
F11 (Save)	Saves all field values you enter. If you leave a panel without saving, any field values you change are lost.
F12 (Cancel)	Ends the current panel and returns you to the panel you were previously on.

Chapter 3. Using the Preprocessor and ITPSYSIN to preprocess WSim scripts

Before you can run your WSim simulation, you must place the network definitions and message generation decks into data sets. You can use either the Preprocessor or the ITPSYSIN utility program to store your network definitions and message generation decks. However, use of the Preprocessor or ITPSYSIN is not required. You can also use an editor to store the network definition statements and message generation decks in data sets.

The difference between the Preprocessor and ITPSYSIN is that the Preprocessor checks the syntax of the statements when it stores them and ITPSYSIN only stores the statements. Because you can also check the syntax when you initialize the network by running WSim, you can use the Preprocessor and ITPSYSIN most efficiently in the following situations:

- Use the Preprocessor:
  - If you have just created the message generation decks
  - If you have made substantial changes to the message generation decks
  - If you are storing the WSim script without initializing and running the simulation.
- Use ITPSYSIN:
  - If you have made only minor changes, for example, to one statement
  - If you are planning on immediately initializing the network and running WSim.

  Note: Using the Preprocessor before initializing the network does not shorten the amount of time needed to initialize a network during WSim execution.

Whether you check your errors by using the Preprocessor or by using ITPSYSIN and initializing the system, you can obtain system output and use it to debug your script.

Using the Preprocessor

The following sections discuss the input required for the Preprocessor and the output that is provided. They also describe how to run the Preprocessor.

Understanding Preprocessor input

The input to the Preprocessor consists of network definition statements, which can be a sequential data set or a member of a partitioned data set, and message generation decks. When preprocessing a network, place the network definition in the input data stream first, followed by the message generation decks. If a member containing a specific message generation deck already exists in the data set specified by the MSGDD DD statement and does not need to be modified, you do not need to include that message deck in the data stream for the Preprocessor. If the network definition already exists in the INITDD DD data set and does not need to be modified, you can use the PREP statement in the input data stream instead of repeating the network definition. Refer to Figure 3 on page 12 for an example of using the PREP statement.

You can include multiple networks in the input stream to be preprocessed during a single Preprocessor run. Include the message generation decks required by these multiple networks in the input stream unless the message generation decks exist in the MSGDD data set. For more information about the syntax of the statements used as input to the Preprocessor, refer to *WSim Script Guide and Reference*.

The PREP statement, which is valid only for the Preprocessor, has the following format:

```
netname   PREP
```

where *netname* is the name of the member from the INITDD data set that is to be preprocessed. If a message generation deck is required by a NTWRK or a PREP statement but is not in the MSGDD DD, you must include it before the next NTWRK or PREP statement. You can code more than one PREP statement. You can also mix the PREP statements with the WSim network definition statements, as shown in Figure 3.

```
TEST1    PREP       BEGIN NETWORK 1 DEFINITION
TEST2    PREP       BEGIN NETWORK 2 DEFINITION
DECK1    MSGTXT
         .
         .
         .
         ENDTXT
TEST3    NTWRK      BEGIN NETWORK 3 DEFINITION
         .
         .
         .

         VTAMAPPL
         LU

         .
         .
         .
DECK2    MSGTXT
         .
         .
         .
         ENDTXT
TEST4    PREP       BEGIN NETWORK 4 DEFINITION
```

Figure 3. Example of Preprocessor input with the PREP statement

Understanding Preprocessor output

If the Preprocessor detects no errors in the network definition statements, the network is stored in the INITDD data set defined in the statements used to run the Preprocessor. The network definition is stored in the output data set defined by the INITDD DD statement under the name that appears on the NTWRK statement. Message generation decks and message user tables are stored in the output data set defined by the MSGDD DD statement under the name that appears on the MSGTXT or MSGUTBL statement. Only the network definition statements and message generation decks obtained from the SYSIN input data stream are written to the respective data sets. Figure 4 on page 13 shows a diagram of the input to and output from the Preprocessor.

Note: The INITDD and MSGDD DD statements can define the same data set. The data sets are stored only when they are preprocessed without an error.

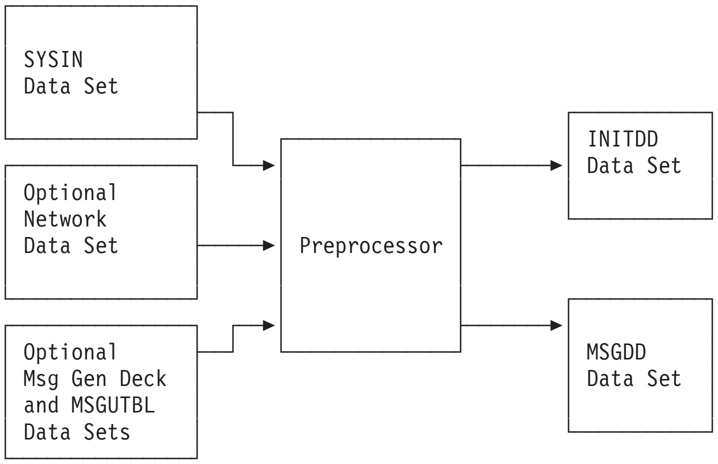

SYSIN Data Set
Optional Network Data Set
Optional Msg Gen Deck and MSGUTBL Data Sets
Preprocessor
INITDD Data Set
MSGDD Data Set

Figure 4. Preprocessor output storage

A preprocessed network run with the PREP statement is not rewritten to the INITDD data set after the initialization process is complete. Message generation decks are processed normally and written to the MSGDD data set only if they were included in the SYSIN data stream.

In addition to data set members, the Preprocessor also provides the following printed output:

- A listing of the input statements (optional)

  The Preprocessor output includes statement numbers of the executable statements of each message generation deck (for all lines except MSGTXT statements, LABEL statements, comments, and blank lines). Each statement number is a 5-digit number printed to the left of the corresponding statement. The message generation decks are each numbered separately beginning with 00001. When a message trace record in the Loglist Utility output refers to a particular statement, you can use this listing to determine what that statement was. Refer to Using the loglist utility to format the log data set for information about using the Loglist Utility.
- A cross-reference report

  Following the statement listing is a cross-reference report, showing each statement, its type, where it is defined, and where it is referenced. See Using the cross-reference report for information about this report.
- A listing of the input errors

  Only the first error detected on each statement line is flagged.
- A network summary report (optional)

  The network summary report summarizes the network definition options and defaults.
- An estimate of storage requirements
- A statement indicating whether the data in the network definition was saved in the INITDD data set.

The following three figures show different types of Preprocessor output for WSim scripts. The first two show output from WSim scripts that had no syntax errors. Figure 5 on page 14 illustrates output without a network summary report and Figure 6 on page 15 illustrates output with a network summary report. All three Preprocessor output listings show a cross-reference report at the end of the listing. Note that the size of the network printed at the end of the Preprocessor run specifies the number of bytes required for the network control blocks; it does not indicate the region size necessary for network execution. The number of text blocks required for a MSGDISK statement also appears in the output.

Figure 7 on page 16 shows Preprocessor output for a network with errors and no network summary report. Note that an error message appears above the incorrect statement, and an asterisk (*) appears below the invalid operand on the scale line.

```
 WSim PREPROCESSOR OUTPUT                  TIME  9.09.01,     APRIL 12, 2002   PAGE      1

    LINE    STMT    ---------1---------2---------3---------4---------5---------6---------7-
       1            VAPPL    NTWRK    ITIME=1,            * Network interval report every 1 *
       2            *                                     * minute                          *
       3                              SCAN=(1,1,0),       * Scan/display/recovery times     *
       4                              UTI=100,            * User time interval = 1 second   *
       5            *-------------------------------------------------------------------------*
       6            * VTAMAPPL operands coded on the network statement.  These values will *
       7            * be the default for every VTAMAPPL in the network.                     *
       8            *-------------------------------------------------------------------------*
       9                              BUFSIZE=3000,       * Buffer size is 3000 bytes       *
      10                              MLOG=YES,           * Message logging function will be*
      11            *                                     * used                            *
      12            *-------------------------------------------------------------------------*
      13            * LU operands coded on the network statement.  These values will be    *
      14            * the default for every Logical Unit in the network.                    *
      15            *-------------------------------------------------------------------------*
      16                              DELAY=F1,           * Use fixed message delay interval*
      17                              DLOGMOD=D4A32782,   * VTAM logon mode                 *
      18                              INIT=SEC,           * Secondary LU initiates the      *
      19            *                                     * session                         *
      20                              LOGDSPLY=BOTH,      * Log the display buffers before  *
      21            *                                     * and after message generation    *
      22                              LUTYPE=LU2,         * Logical unit type = LU2         *
      23                              MSGTRACE=YES,       * Write message generation trace  *
      24            *                                     * records to the log              *
      25                              PATH=(0),           * Specifies which PATH statement  *
      26            *                                     * the LU will use                 *
      27                              THKTIME=UNLOCK      * Terminal unlock starts msg delay*
      28            *                                     * timer                           *
      29            *-------------------------------------------------------------------------*
      30            0        PATH     LUDECK              * Run the LUDECK msgtxt on this   *
      31            *                                     * path                            *
      32            *-------------------------------------------------------------------------*
      33            * Define the network resources.                                         *
      34            *                                                                       *
      35            * ==> CHANGE the VTAMAPPL names APPL1 and APPL2 as needed to match      *
      36            *     names in your environment.  These names must be defined to VTAM. *
      37            *-------------------------------------------------------------------------*
      38            APPL1    VTAMAPPL
      39            LU11     LU
      40            APPL2    VTAMAPPL
      41            LU21     LU
      42            *
      43            LUDECK   MSGTXT
      44            *************************************************************************
      45            * The Message Generation deck.                                          *
      46            * Issue an INITSELF to log on to ITPECHO.                               *
      47            * Wait until the logon is complete.                                     *
      48            * After the logon completes, issue a message to the console.           *
      49            *************************************************************************
      50       1             CMND COMMAND=INIT,RESOURCE=ITPECHO
      51       2    0        IF LOC=RU+0,TEXT=(WELCOME),SCAN=YES,THEN=CONT,ELSE=WAIT
      52       3             WAIT
      53       4             WTO ($LUID$ UP AND RUNNING)
      54            *************************************************************************
      55            * Send a message to ITPECHO.                                            *
      56            * Repeat the loop forever until the WSim operator stops the network.   *
      57            *************************************************************************
      58            LOOP     LABEL
      59       5             TEXT (MSG $DSEQ,5$ FROM $LUID$ )
      60       6             BRANCH LABEL=LOOP
      61       7             ENDTXT
CROSS-REFERENCE REPORT                     TIME  9.09.01,     APRIL 12, 2002   PAGE      2
 NAME     (DECK)      TYPE         DEFINED    REFERENCED ON LINE NUMBER
 -------------------  -----------  -------    ---------------------------------------------
 DSEQ                 COUNTER                 59
 LOOP     (LUDECK)    LABEL             58    60
 LUDECK               DECK NAME         43    30
 NTWRKUTI             UTI NAME           4
 RU                   DATA STREAM             51
 0                    IF#                     51
 ITP657I         25,144 BYTES ARE REQUIRED FOR THIS NETWORK
 ITP659I           1 BLOCKS OF TEXT DATA ARE REQUIRED FOR THIS NETWORK
 ITP652I INITDD VAPPL    REPLACED IN DATA SET
 ITP652I MSGDD  LUDECK   REPLACED IN DATA SET
```

Figure 5. Example of Preprocessor output with no errors

```
 WSim PREPROCESSOR OUTPUT                  TIME  9.08.28,     APRIL 12, 2002   PAGE      1

    LINE    STMT    ---------1---------2---------3---------4---------5---------6---------7-
       1            VAPPL    NTWRK    ITIME=1,           * Network interval report every 1 *
       2            *                                    * minute                          *
       3                              SCAN=(1,1,0),      * Scan/display/recovery times     *
       4                              UTI=100,           * User time interval = 1 second   *
       5            *----------------------------------------------------------------------*
       6            * VTAMAPPL operands coded on the network statement.  These values will *
       7            * be the default for every VTAMAPPL in the network.                    *
       8            *----------------------------------------------------------------------*
       9                              BUFSIZE=3000,      * Buffer size is 3000 bytes       *
      10                              MLOG=YES,          * Message logging function will be*
      11            *                                    * used                            *
      12            *----------------------------------------------------------------------*
      13            * LU operands coded on the network statement.  These values will be    *
      14            * the default for every Logical Unit in the network.                   *
      15            *----------------------------------------------------------------------*
      16                              DELAY=F1,          * Use fixed message delay interval*
      17                              DLOGMOD=D4A32782,  * VTAM logon mode                 *
      18                              INIT=SEC,          * Secondary LU initiates the      *
      19            *                                    * session                         *
      20                              LOGDSPLY=BOTH,     * Log the display buffers before  *
      21            *                                    * and after message generation    *
      22                              LUTYPE=LU2,        * Logical unit type = LU2         *
      23                              MSGTRACE=YES,      * Write message generation trace  *
      24            *                                    * records to the log              *
      25                              PATH=(0),          * Specifies which PATH statement  *
      26            *                                    * the LU will use                 *
      27                              THKTIME=UNLOCK     * Terminal unlock starts msg delay*
      28            *                                    * timer                           *
      29            *----------------------------------------------------------------------*
      30            0        PATH     LUDECK             * Run the LUDECK msgtxt on this   *
      31            *                                    * path                            *
      32            *----------------------------------------------------------------------*
      33            * Define the network resources.                                        *
      34            *                                                                      *
      35            * ==> CHANGE the VTAMAPPL names APPL1 and APPL2 as needed to match     *
      36            *     names in your environment.  These names must be defined to VTAM. *
      37            *----------------------------------------------------------------------*
      38            APPL1    VTAMAPPL
      39            LU11     LU
      40            APPL2    VTAMAPPL
      41            LU21     LU
      42            *
      43            LUDECK   MSGTXT
      44            ************************************************************************
      45            * The Message Generation deck.                                         *
      46            * Issue an INITSELF to log on to ITPECHO.                              *
      47            * Wait until the logon is complete.                                    *
      48            * After the logon completes, issue a message to the console.           *
      49            ************************************************************************
      50       1             CMND COMMAND=INIT,RESOURCE=ITPECHO
      51       2    0        IF LOC=RU+0,TEXT=(WELCOME),SCAN=YES,THEN=CONT,ELSE=WAIT
      52       3             WAIT
      53       4             WTO ($LUID$ UP AND RUNNING)
      54            ************************************************************************
      55            * Send a message to ITPECHO.                                           *
      56            * Repeat the loop forever until the WSim operator stops the network.   *
      57            ************************************************************************
      58            LOOP     LABEL
      59       5             TEXT (MSG $DSEQ,5$ FROM $LUID$ )
      60       6             BRANCH LABEL=LOOP
      61       7             ENDTXT
CROSS-REFERENCE REPORT                      TIME  9.08.28,     APRIL 12, 2002   PAGE      2
 NAME     (DECK)      TYPE         DEFINED    REFERENCED ON LINE NUMBER
 -------------------  -----------  -------    ---------------------------------------------
 DSEQ                 COUNTER                 59
 LOOP     (LUDECK)    LABEL             58    60
 LUDECK               DECK NAME         43    30
 NTWRKUTI             UTI NAME           4
 RU                   DATA STREAM             51
 0                    IF#                     51
 WSim PREPROCESSOR OUTPUT                                           TIME  9.08.28,     APRIL 12, 2002   PAGE      3
                                                      NETWORK SUMMARY
 VAPPL     NTWRK      HEADING='  WSim INTERVAL REPORT  '   UTI=100      REPORT=FULL   ITIME=1     STIME=0
                     SCAN=(1,1,0)                 EMTRATE NOT ACTIVE      NAMEHASH=10
                     NETUSER=0                   CNTRS=7    OPTIONS NOT CODED
                     CNTRSEED=7935629      DELYSEED=9104901      PATHSEED=1532001      TEXTSEED=3841995      UTBLSEED=5736539
                     INEXIT NOT CODED   OUTEXIT NOT CODED   NCTLEXIT NOT CODED   UCMDEXIT NOT CODED   UXOCEXIT NOT CODED
                                          INFOEXIT NOT CODED
                     INXEXPND=NO
                     INHBTMSG NOT CODED
 APPL1      VTAMAPPL       APPLID=APPL1      BUFSIZE=3000       MLEN=ALL     MLOG=YES   PASSWD=APPL1
       LU11-1        LU    LUTYPE=LU2                                  THKTIME=UNLOCK   DELAY=F(1)                    QUIESCE=NO
                           IUTI=NTWRKUTI   THROTTLE=1
                           FRSTTXT NOT CODED   ATRDECK NOT CODED   SAVEAREA=(0,0)        USERAREA=0
                           ATRABORT=DECK   MAXCALL=5     SEQ=0                     DISPLAY=(24,80,24,80)
                           LOGDSPLY=BOTH   PROTMSG=YES
                           EXTFUN=YES   COLOR=GREEN    HIGHLITE=NO    FLDVALID=NO    MAXNOPTN=0     MAXPTNSZ=0
                           PS=NONE            ALTCSET=NONE   CCSIZE=(9,16)      UOM=INCH   UASIZE=(960,751)
                           DBCS=NO    FLDOUTLN=NO    BASECSID=(697,37)
                           MAXSESS=(0,1)         INIT=SEC   RESOURCE NOT CODED   RTR=NO    ENCR=NONE   CHAINING N/A
                           CRDATALN=20   MSGTRACE=YES   STLTRACE=NO    PATH=(0)   RSTATS=NO    DLOGMOD=D4A32782
 APPL2      VTAMAPPL       APPLID=APPL2      BUFSIZE=3000       MLEN=ALL     MLOG=YES   PASSWD=APPL2
       LU21-1        LU    LUTYPE=LU2                                  THKTIME=UNLOCK   DELAY=F(1)                    QUIESCE=NO
                           IUTI=NTWRKUTI   THROTTLE=1
                           FRSTTXT NOT CODED   ATRDECK NOT CODED   SAVEAREA=(0,0)        USERAREA=0
                           ATRABORT=DECK   MAXCALL=5     SEQ=0                     DISPLAY=(24,80,24,80)
                           LOGDSPLY=BOTH   PROTMSG=YES
                           EXTFUN=YES   COLOR=GREEN    HIGHLITE=NO    FLDVALID=NO    MAXNOPTN=0     MAXPTNSZ=0
                           PS=NONE            ALTCSET=NONE   CCSIZE=(9,16)      UOM=INCH   UASIZE=(960,751)
                           DBCS=NO    FLDOUTLN=NO    BASECSID=(697,37)
                           MAXSESS=(0,1)         INIT=SEC   RESOURCE NOT CODED   RTR=NO    ENCR=NONE   CHAINING N/A
                           CRDATALN=20   MSGTRACE=YES   STLTRACE=NO    PATH=(0)   RSTATS=NO    DLOGMOD=D4A32782
 ITP657I         25,144 BYTES ARE REQUIRED FOR THIS NETWORK
 ITP659I           1 BLOCKS OF TEXT DATA ARE REQUIRED FOR THIS NETWORK
 ITP652I INITDD VAPPL    REPLACED IN DATA SET
 ITP652I MSGDD  LUDECK   REPLACED IN DATA SET
```

Figure 6. Example of Preprocessor output using the SUMMARY option

```
WSim PREPROCESSOR OUTPUT                   TIME  9.09.43,     APRIL 12, 2002   PAGE      1

   LINE    STMT    ---------1---------2---------3---------4---------5---------6---------7-
      1            VAPPL    NTWRK    ITIME=1,           * Network interval report every 1 *
      2            *                                    * minute                          *
      3                              SCAN=(1,1,0),      * Scan/display/recovery times     *
      4                              UTI=100,           * User time interval = 1 second   *
      5            *----------------------------------------------------------------------*
      6            * VTAMAPPL operands coded on the network statement.  These values will *
      7            * be the default for every VTAMAPPL in the network.                    *
      8            *----------------------------------------------------------------------*
      9                              BUFSIZE=3000,      * Buffer size is 3000 bytes       *
     10                              MLOG=YES,          * Message logging function will be*
     11            *                                    * used                            *
     12            *----------------------------------------------------------------------*
     13            * LU operands coded on the network statement.  These values will be    *
     14            * the default for every Logical Unit in the network.                   *
     15            *----------------------------------------------------------------------*
     16                              DELAY=F1,          * Use fixed message delay interval*
     17                              DLOGMOD=D4A32782,  * VTAM logon mode                 *
     18                              INIT=SEC,          * Secondary LU initiates the      *
     19            *                                    * session                         *
     20                              LOGDSPLY=BOTH,     * Log the display buffers before  *
     21            *                                    * and after message generation    *
     22                              LUTYPE=LU2,        * Logical unit type = LU2         *
     23                              MSGTRACE=YES,      * Write message generation trace  *
     24            *                                    * records to the log              *
     25                              PATH=(0),          * Specifies which PATH statement  *
     26            *                                    * the LU will use                 *
INITIALIZATION ERROR
VAPPL               NEAR COLUMN 27     ITP1211I    OPERAND VALUE IS INVALID
     27                              THKTIME=UNLCK      * Terminal unlock starts msg delay*
                   ----+----1----+----2----+-*--3----+----4----+----5----+----6----+----7----+----8
     28            *                                    * timer                           *
     29            *----------------------------------------------------------------------*
     30            0        PATH     LUDECK             * Run the LUDECK msgtxt on this   *
     31            *                                    * path                            *
     32            *----------------------------------------------------------------------*
     33            * Define the network resources.                                        *
     34            *                                                                      *
     35            * ==> CHANGE the VTAMAPPL names APPL1 and APPL2 as needed to match     *
     36            *     names in your environment.  These names must be defined to VTAM. *
     37            *----------------------------------------------------------------------*
     38            APPL1    VTAMAPPL
     39            LU11     LU
     40            APPL2    VTAMAPPL
     41            LU21     LU
     42            *
     43            LUDECK   MSGTXT
     44            ************************************************************************
     45            * The Message Generation deck.                                         *
     46            * Issue an INITSELF to log on to ITPECHO.                              *
     47            * Wait until the logon is complete.                                    *
     48            * After the logon completes, issue a message to the console.           *
     49            ************************************************************************
     50       1             CMND COMMAND=INIT,RESOURCE=ITPECHO
     51       2    0        IF LOC=RU+0,TEXT=(WELCOME),SCAN=YES,THEN=CONT,ELSE=WAIT
     52       3             WAIT
     53       4             WTO ($LUID$ UP AND RUNNING)
     54            ************************************************************************
     55            * Send a message to ITPECHO.                                           *
     56            * Repeat the loop forever until the WSim operator stops the network.   *
     57            ************************************************************************
INITIALIZATION ERROR
VAPPL    LUDECK     NEAR COLUMN 02     ITP1206I INVALID NAME FIELD FOR THIS STATEMENT
     58       5    LOOPUNTILLABEL
                   -*--+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
     59       6             TEXT (MSG $DSEQ,5$ FROM $LUID$ )
     60       7             BRANCH LABEL=LOOP
     61       8             ENDTXT
INITIALIZATION ERROR
VAPPL               NEAR COLUMN 02     ITP1293I LABEL PRINTED NOT FOUND IN MESSAGE DECK LUDECK
                    LOOP
                   -*--+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
CROSS-REFERENCE REPORT                              TIME  9.09.43,     APRIL 12, 2002   PAGE      2
NAME     (DECK)      TYPE         DEFINED    REFERENCED ON LINE NUMBER
-------------------  -----------  -------    -------------------------------------------------------
DSEQ                 COUNTER                 59
LOOP     (LUDECK)    LABEL        *ERROR*    60
LUDECK               DECK NAME         43    30
NTWRKUTI             UTI NAME           4
RU                   DATA STREAM             51
0                    IF#                     51
```

Figure 7. Example of Preprocessor output with errors

Using the cross-reference report

The cross-reference report, printed at the end of the Preprocessor listing, alphabetically lists message generation deck resources used in the script, their types (such as buffer or user area, for example), where they are defined, and where they are used. You can use this report to help debug your script.

Figure 8 shows an example of this report.

```
NAME     (DECK)      TYPE        DEFINED    REFERENCED ON LINE NUMBER
-------------------  ----------- -------    ----------------------------

EVENT1               ON EVENT               101, 130, 400, 421, 534,
                                            900, 1001, 1200, 1230

EVENT2               WAIT EVENT             102, 150, 172

LOGOFFDK             DECK NAME        40    10, 100

XITLABEL (LOGOFFDK)  LABEL            45    110, 217
```

Figure 8. Cross-reference report

Reading the cross-reference report

The cross-reference report contains four columns of information:

- NAME
- TYPE
- DEFINED
- REFERENCED ON LINE NUMBER.

The NAME column lists the names of all message deck resources used. If the resource is a label, the name of the message deck where it is defined is shown in parentheses.

The TYPE column lists the types of resources used. Valid values for the TYPE column are:

BUFFER	Device buffer (B). **Note:** The default for the LOC operand on the DATASAVE statement is B (buffer). The report references BUFFER when this default is taken. Also, the report references LOC=* on the DATASAVE statement as BUFFER.
COUNTER	Sequence or index counter.
CURSOR	Cursor position (C).
DATA STREAM	Data stream entities (TH, RH, RU, and D).
DECK NAME	Message generation deck name.
EVENT TAG	Event tag name. **Note:** The default for the EVENTTAG operand when TIME is specified on the EVENT statement is the event name. The report references the event name as an EVENT TAG when this default is taken.
IF#	Message generation deck IF statement number. **Note:** When DEACT IFS=ALL is coded, an entry of *ALL* appears in the NAME column for this item, indicating all IFs are deactivated.
LABEL	Statement label. The message deck containing the label definition is also shown, in parentheses, in the name column.

ON EVENT	ON/SIGNAL event name. **Note:** When DEACT ONEVENTS=ALL is coded, an entry of *ALL* appears in the NAME column for this item, indicating all ON events are deactivated.
SAVE AREA	Save area number. **Note:** The default for the AREA operand on the DATASAVE statement is save area 1. The report references SAVE AREA 1 when this default is taken.
SWITCH	Network, terminal, or device switch.
USER AREA	User area number.
UTBL	UTBL or MSGUTBL statement.
UTI NAME	UTI statement name.
WAIT EVENT	WAIT/POST event name.

The DEFINED column is used only for items that must be uniquely defined. These are deck name, statement label, user table, and UTI name. If an item that is supposed to be defined is not, the report displays *ERROR* in the column. If one of these items is defined more than once, only the first location's line number appears in the report.

The REFERENCED ON LINE NUMBER column lists the line numbers containing statements referencing that resource.

Note: The cross-reference report may not be correct if Preprocessor errors are found.

Running the Preprocessor

To run the Preprocessor, use the WSim load module ITPENTER with the execution parameter PREP specified. The following sections describe the execution parameters used to run the Preprocessor and give a sample JCL and TSO CLIST to run the preprocessor. You can also run the Preprocessor from the WSim/ISPF Interface.

Note: During the Preprocessor run, most of the control blocks necessary to execute the network are built in virtual storage just as if the network were to be executed. Therefore, when preprocessing large networks, the region size specified for the job must be large enough for the WSim code, the network control blocks, and the message generation decks. For more information about determining the region size, refer to the virtual storage estimates section in *WSim User's Guide*.

Using the WSim/ISPF Interface

You can run the Preprocessor from the WSim/ISPF Interface. To do this, follow these steps:

1. Invoke the WSim/ISPF Interface main panel from ISPF. The method you use to do this depends on how the WSim/ISPF Interface application is installed at your site. If you are not sure how to do this, see your system programmer for assistance.
2. Select option 2 from the WSim/ISPF Interface main panel and press **Enter**. The Preprocess Networks and Message Decks panel is displayed.

   Note: You can also type “PREP” on the command line and press **Enter** to display this panel.
3. Fill in the appropriate information on this panel and press **Enter** to preprocess your script.

For more information on the WSim/ISPF Interface, see Chapter 2, “Running WSim with the WSim/ISPF Interface,” on page 5.

Using Preprocessor execution parameters

You can enter the following execution parameters in the PARMDD data set on the EXEC statement (MVS) or on the CALL statement for TSO CLISTs when you run the Preprocessor. The PARMDD data set parameters are processed first.

ADD
: Specifies that all members of a network preprocessed with no errors are to be added to the output data sets (INITDD and MSGDD) only if the data sets do not already contain members with the same names. If a member already exists with the same name, it will not be replaced. If there are any errors in the network, no attempt is made to add the data to the output data set. ADD is the default.

  Refer to execution parameter REPL for information about replacing members.

NOLIST
: Specifies that a network listing is not to be printed in the SYSPRINT data set. Any errors detected will still be printed.

NOXREF
: Specifies that a cross-reference report is not to be printed.

PREP
: Specifies that WSim is to be executed for the purpose of preprocessing networks. PREP is required in order to invoke the Preprocessor.

PRTLNCNT=*nnn*
: Specifies the maximum number of lines to be printed on a page of output before ejecting to a new page. Enter an integer from 35 to 255 for *nnn*. The default value for *nnn* is 60.

REPL
: Specifies that all members of a network preprocessed with no errors are to be added to the output data sets, replacing any members that already exist by the same names. If there are any errors in the network, no attempt is made to replace or add the data to the output data set.

  Refer to the execution parameter ADD for information about adding members to the data set.

SUMMARY
: Specifies that a report summarizing the network definition options and defaults is to be printed in the SYSPRINT data set.

XREF
: Specifies that a cross-reference report is to be printed. If you do not specify either XREF or NOXREF, a cross-reference report is printed.

Using JCL

The following shows sample JCL to run the Preprocessor on MVS.

```
//PREPJOB  JOB  MSGLEVEL=1
//STEP1    EXEC PGM=ITPENTER,PARM='PREP,ADD,NOLIST',REGION=2048K
//STEPLIB  DD   DSN=WSIM.SITPLOAD,DISP=SHR
//PARMDD   DD   DSNAME=WSIM.PARMDD,DISP=SHR
//RATEDD   DD   DSNAME=WSIM.SITPRTBL,DISP=SHR
//INITDD   DD   DSNAME=WSIM.TESTFILE,DISP=SHR
//MSGDD    DD   DSNAME=WSIM.MSGFILE,DISP=SHR
//SYSUT2   DD   UNIT=SYSDA,SPACE=(TRK,(10,10,3))
//SYSUT3   DD   UNIT=SYSDA,SPACE=(TRK,(10,10,3))
```

```
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
TESTNET    NTWRK
      .
      .
      .
/*
```

The following descriptions explain the JCL statements shown above.

Statement	Function
PREPJOB JOB	Initiates the job.
STEP1 EXEC	Specifies the program name, ITPENTER, and the PREP execution parameter. Refer to "Using Preprocessor execution parameters" on page 19 for more information about the optional parameters, such as ADD and NOLIST.
STEPLIB DD	Defines the data set containing the WSim host processor load modules.
PARMDD DD	Defines an optional sequential data set containing ITPENTER execution parameters. The following syntax rules apply to the records in this data set. • An asterisk (*) in column 1 denotes a comment record. • One or more parameters may be coded on each record, delimited by commas. • Any data following a trailing blank is considered a comment. • Leading blanks are allowed. • A trailing comma is not required to indicate continuation of parameters on the next record. The BLKSIZE for this data set must be a multiple of 80. This statement is optional.
RATEDD DD	Defines the RATE tables file. This statement is required only if your network contains a RATE statement.
INITDD DD	Defines a partitioned data set to receive the network being processed. The data set can already contain the network to be processed when using the PREP statement, and it can be the same as the MSGDD data set. The BLKSIZE for this data set must be a multiple of 80. This statement is required.
MSGDD DD	Defines a partitioned data set to receive the message generation decks being processed. The data set may already contain the message generation decks to be processed, and it can be the same as the INITDD data set. The BLKSIZE for this data set must be a multiple of 80. This statement is required.
SYSUT2 DD	Defines a partitioned data set that the Preprocessor will use as work space for storing network definition statements. The data set should be large enough to contain all networks being preprocessed. This statement is required if there are any NTWRK statements in the SYSIN data set. This statement is *not* required if all networks are named by PREP statements.
SYSUT3 DD	Defines a partitioned data set that the Preprocessor will use as work space for storing the message generation decks. The data set should be large enough to contain all message generation decks being processed. It must not be the same data set defined by SYSUT2. This statement is required if there are any MSGTXT or MSGUTBL statements in the SYSIN data set.

Statement	Function
SYSPRINT DD	Defines the printer output. If you omit this statement, WSim allocates this data set dynamically.
SYSIN DD	Defines your input data (network definition statements and message generation decks). This statement is required.

Using a TSO CLIST

The following example shows a CLIST used to run the Preprocessor under TSO. An explanation of these statements can be found in "Using JCL" on page 19.

```
ALLOC DDNAME(SYSPRINT) SYSOUT(A)
ALLOC DDNAME(INITDD) DSNAME('WSIM.TESTFILE') SHR
ALLOC DDNAME(MSGDD) DSNAME('WSIM.MSGFILE') SHR
ALLOC DDNAME(RATEDD) DSNAME('WSIM.SITPRTBL') SHR
ALLOC DDNAME(PARMDD) DSNAME('WSIM.PARMDD') SHR
ATTRIB TEMPLST BLKSIZE(800) DSORG(PO)
ALLOC DDNAME(SYSUT2) NEW SPACE(5,5) CYLINDERS DELETE UNIT(SYSDA)
      DIR(3) USING(TEMPLST)
ALLOC DDNAME(SYSUT3) NEW SPACE(5,5) CYLINDERS DELETE UNIT(SYSDA)
      DIR(3) USING(TEMPLST)
ALLOC DDNAME(SYSIN) DSNAME('USER.NETNAME.DATA') SHR
CALL 'WSIM.SITPLOAD(ITPENTER)' 'PREP,REPL,SUMMARY'
FREE DDNAME(SYSPRINT)
FREE DDNAME(INITDD)
FREE DDNAME(MSGDD)
FREE DDNAME(RATEDD)
FREE DDNAME(PARMDD)
FREE DDNAME(SYSUT2)
FREE DDNAME(SYSUT3)
FREE DDNAME(SYSIN)
```

Understanding Preprocessor return codes

After running, the Preprocessor sets a return code to indicate the status of the execution. The Preprocessor can return the following codes:

Code	Meaning
0	The run completed with no errors.
4	Storage was not available for preprocessor execution. The preprocessor stops.
8	OPEN failed for one of the input, output, or work data control blocks, or ATTACH failed for the initiator subtask. The preprocessor stops.
12	An invalid execution parameter was specified. The preprocessor stops.
16	Message generation decks were entered without a network definition, and the decks were not processed.
20	Network initialization ended with errors for at least one of the networks in the input data stream.
24	A STOW (partitioned data set directory update) failed for one of the output data sets. The preprocessor stops.
36	Error reading data from the SYSIN input data set. The preprocessor stops.

Using ITPSYSIN

You can use a subset of the Preprocessor functions by running the ITPSYSIN utility program. ITPSYSIN stores message generation decks and network definition statements in the appropriate data set without checking syntax. ITPSYSIN runs much faster than the Preprocessor. If you plan to use WSim immediately after you run the Preprocessor and are sure you did not make a syntax error:

- Use ITPSYSIN instead of the preprocessor
- Initialize the network with the list option to get a listing of the network.

This way, the syntax of the statements in the network is checked only once (when initialized) instead of twice.

You can use ITPSYSIN any time you do not need the syntax checking facilities of the Preprocessor. For example, you could use ITPSYSIN for a personalized library system, previously preprocessed networks, automatically generated networks, or any other situations where you already know the syntax is correct.

Note: When using ITPSYSIN, remember that the syntax of the networks will still be checked when they are initialized. You must correct all errors before WSim can execute a network.

Understanding ITPSYSIN input and output

ITPSYSIN reads the SYSIN data stream and stores the network definition statements and message generation decks in the data sets defined by the INITDD and MSGDD DD statements, respectively. ITPSYSIN does not check the syntax of the network definition statements or message generation decks.

Running ITPSYSIN

The following sections give examples of JCL and a TSO CLIST to run ITPSYSIN, as well as how to run ITPSYSIN from the WSim/ISPF Interface.

Using the WSim/ISPF Interface

To run ITPSYSIN from the WSim/ISPF Interface, follow these steps:

1. Invoke the WSim/ISPF Interface main panel from ISPF. The method you use to do this depends on how the WSim/ISPF Interface application is installed at your site. If you are not sure how to do this, see your system programmer for assistance.
2. Select option 2 from the WSim/ISPF Interface main panel and press **Enter**. The Preprocess Networks and Message Decks panel is displayed.

   Note: You can also type “PREP” on the command line and press **Enter** to display this panel.
3. Fill in the appropriate information on this panel.
4. Type “N” in the Check syntax field.
5. Press **Enter** to run the utility.

For more information on the WSim/ISPF Interface, see Chapter 2, “Running WSim with the WSim/ISPF Interface,” on page 5.

Using JCL

The following shows sample JCL to run ITPSYSIN on MVS.

```
//SYSINJOB JOB  MSGLEVEL=1
//JOBLIB   DD   DSN=WSIM.SITPLOAD,DISP=SHR
//STEP1    EXEC PGM=ITPSYSIN
//INITDD   DD   DSNAME=WSIM.TESTFILE,DISP=SHR
//MSGDD    DD   DSNAME=WSIM.MSGFILE,DISP=SHR
//SYSIN    DD   *
TESTNET    NTWRK
      .
      .
      .
/*
```

Using a TSO CLIST

The following is an example of a CLIST to run ITPSYSIN under TSO.

```
ALLOC  DDNAME(INITDD) DATASET('WSIM.TESTFILE') SHR
ALLOC  DDNAME(MSGDD) DATASET('WSIM.MSGFILE') SHR
ALLOC  DDNAME(SYSIN) DATASET('USER.NETNAME.DATA') SHR
CALL   'WSIM.SITPLOAD(ITPSYSIN)'
FREE   DDNAME(INITDD MSGDD SYSIN)
```

Understanding ITPSYSIN return codes

After running, ITPSYSIN sets a return code to indicate the status of the execution. ITPSYSIN can return the following codes:

Code	Meaning
0	The run completed with no errors.
4	The SYSIN data set failed to open.
8	The MSGDD data set failed to open.
16	Storage was not available for ITPSYSIN execution.
20	A STOW (partitioned data set directory update) failed for the networks or message decks.

Refer to *WSim Messages and Codes* for informational and error messages that originate from the preprocessor and ITPSYSIN.

Chapter 4. Using the Loglist Utility to format the log data set

The Loglist Utility produces formatted reports of the activity recorded in the WSim log data set. You can specify a report for all the resources in the WSim log data set or for specific network resources. For information about message logging, time stamping messages, and how data messages are written to the log data set, refer to Chapter 12, “Understanding message logging,” on page 163.

The Loglist Utility has two input sources: the WSim log data set and a group of control statements. The log data set is typically stored on a tape or disk, and the control statements may be provided either from a file or in response to write-to-operator-with-reply (WTOR) messages at the console. You can run the Loglist Utility using JCL, a TSO CLIST, or from the WSim/ISPF Interface.

Before you use the Loglist Utility, you need to decide which type of reports and records you want to produce. To do that, you need to know the meanings of the headers on the reports. Then, to run the Loglist Utility, you need to use control commands. The sections in this chapter present information about the following items:

- The types of records and reports you can obtain with the Loglist Utility
- The Loglist Utility output header
- The Loglist Utility display attribute table header
- How to run the Loglist Utility, including:
  - Running the Loglist Utility from the WSim/ISPF Interface
  - Execution parameters to use with JCL or a TSO CLIST
  - Examples of JCL and a TSO CLIST for running the Loglist Utility
  - Sample output from an example of an input command file
  - Return codes after running the Loglist Utility.

Information you can obtain with the Loglist Utility

This section describes the different log data set records that you can format with the Loglist Utility.

SNA resource records

For SNA resource records, if you use the NOFMT control command for the Loglist Utility, all transmit and receive records for simulated SNA resources are printed only in hexadecimal characters. Otherwise, they are formatted in the output listing. The formatted output breaks each SNA transaction into its logical components and describes the meaning of the bits and data fields in the components.

The first line printed after the log header identifies the transaction as an SNA request or response that was transmitted or received. If the request or response unit (RU) contains a command, the command name is printed.

The next three output lines format the transaction into the Data Link Control (DLC) fields, the transmission header (TH), and the request or response header (RH).

Note: The RH line is not present if the transaction is a middle or last segment.

The next output line is the request/response unit (RU), printed in hexadecimal characters and then translated to EBCDIC as in a storage dump. The only RU that is formatted into words is the BIND command.

NS (Network Services) RUs that flow between the WSim SNA layer and the VTAM/WSim subtask interface layer are identified by a plus sign in the first column of the record type field in the log record header (+XMIT or +RECV). This indicates that the RU cannot be sent or did not originate from VTAM and was processed or generated by the WSim/VTAM subtask interface layer.

Refer to *WSim User's Guide* for more specific information about when messages are logged.

TCP/IP resource records

You can include some special information and log records for simulated TCP/IP resources.

For Telnet 3270 simulations that include OPTIONS=(DEBUG) in the NTWRK definition, Telnet negotiation options and partial 3270 data buffers are included in the loglist output with a %RECV record header. A single 3270 data buffer may arrive in pieces. Coding the DEBUG option causes each piece to log separately with a %RECV log record header. Once all pieces are received, the complete 3270 data buffer is logged as usual.

For FTP clients, the header information indicates whether the logged data flows on the command connection or data connection. Commands that control file transfer flow on the command connection, and actual file data flows on the data connection. Commands entered by the simulated FTP user to be interpreted by the WSim FTP simulation are logged as %XMIT records. Commands and data built by WSim and transmitted to the FTP server are logged as normal XMIT records. Information actually received from the server is logged as RECV records, but messages built by WSim and passed to the simulated client as if received are logged as %RECV records.

Messages flowing on the command connection are logged in EBCDIC even though they flow in ASCII. File data on the data connection is logged as it flows. If WSim is aware that thefile data is ASCII data; the data is interpreted accordingly in the character portion of the loglist output.

For Simple TCP or UDP client simulation, all data sent is logged as XMIT records and all data received is logged as RECV records. In addition, a null message (XMIT or RECV record with no data) is logged when a connection to a remote host is broken. An XMIT null record indicates that the breaking of the connection was instigated by the WSim script. A RECV null message indicates that the connection was broken at the instigation of the server.

Loglist data output and display

The WSim loglist data output for Simple TCP and Simple UDP is formatted as ASCII and EBCDIC data with 24 bytes of hex data with ASCII and EBCDIC interpretations. ASCII data will be framed with '<' and '>' characters.

Informational records

An informational record is written to the log data set for a particular resource when an error occurs during a simulation, when WSim accomplishes an action or state change, or when a user exit routine starts logging data for the WSim exit interface. The data is printed immediately following the formatted log header. EBCDIC data is printed as is, and hexadecimal data is printed in hexadecimal characters and translated to EBCDIC as in a storage dump.

LOG records

You can use the LOG statement to write WSim control blocks and data areas to the log data set. WSim uses the following headers to describe the control blocks and data areas when the LOG record is formatted:

SAVEAREA (Followed by a number)
Indicates that a dump of the specified device or network save area follows.

USERAREA
Indicates that a dump of the device user area follows.

NETWORK USERAREA
Identifies the record as a dump of the network user area.

SEQUENCE AND INDEX COUNTERS
Indicates a dump of the sequence counter followed by the index counters for each of the resource levels according to the following order: network, terminal, and device or logical unit.

NETWORK AND DEVICE SWITCHES
Indicates a dump of the 4095 network-level switches followed by the 4095 switches for the current device or logical unit. The leftmost bit of the network switches is NSW1 and the rightmost bit is NSW4095. The leftmost bit of the device switches is SW1 and the rightmost bit is SW4095.

The following headers identify dumps of WSim control blocks:

DEV	Device or logical unit control block
NCB	Network control block
TRM	Terminal control block.

Message trace records

If the MSGTRACE option for a simulation run was specified in your network definition statement or with the A (Alter) command, messages providing data about the logic tests will appear on the log data set. A message trace record will be present for each logic test for which execution was attempted for each device. The Loglist Utility does not list these message trace records individually; the records are grouped by device name for a single pass through the logic test routine. That is, a single group of message trace (MTRC) records on the Loglist Utility output report will correspond to a single message transmission or receipt for a particular device. For transmit messages, the set of message trace records for the logic tests on the transmitted data will be listed on the output report *before* the actual transmit message. For receive messages, the set of message trace records for the logic tests on the received data will be listed on the output report *after* the actual received message.

STL trace records

The STL trace facility, like the message trace facility, traces the messages of a WSim simulation; however, the STL facility traces activity at the STL program level. To use the STL facility, you must specify the STLTRACE option for a simulation run in your network definition or with the A (Alter) operator command. You must also code an @PROGRAM statement in your STL program or specify the PROGRAM execution parameter and not specify NOPDSOUT when you run the STL Translator. WSim will log the STL trace records to the log data set, enabling you to uncover errors in program logic when you format the records using the Loglist Utility. Refer to *WSim Script Guide and Reference* for information about STL programs and coding STL statements.

CPI-C trace records

WSim logs CPI-C trace records using the CTRC record type. The CPITRACE operand on a TP network definition statement specifies the level of CPI-C tracing to be performed. Either the CPI-C verbs themselves, or messages tracing the CPI-C verb flows, can be written to the log data set for formatting by the WSim loglist program. If CPITRACE=VERB, VERBEND, or MSG, information providing data about the CPI-C verbs will appear on the log data set. For more information about the levels of CPI-C tracing, refer to *WSim Script Guide and Reference*.

WSim logs CPI-C attach requests and send and receive data as XMIT and RECV records. The MLOG network definition operand controls the logging of these records. You can also use the Response Time Utility to analyze these records.

Verification reports

When coded for a Loglist Utility run, the VERIFY command causes all VRFY log records to be processed within the boundaries of the run as specified by other Loglist Utility commands. VRFY record processing consists of two types of reports:

- Verification Detail Reports
- Verification Summary Report.

The following sections describe both types of verification reports.

Verification Detail Reports

Verification Detail Reports appear in the Loglist Utility output wherever one or more VRFY records are encountered. These reports are interspersed with other Loglist Utility output records. Figure 9 shows the format of a Verification Detail Report.

```
                             VERIFICATION REPORT
     DESCRIPTION        LOCATION    LENG COND        EXPECTED VALUE            ACTUAL VALUE
======================= =========== ===== == ============================= ===================
SA1 OK                  1+0            11 EQ SAVE AREA 1                   SAVE AREA 1
N4095 OK                N4095+0        18 EQ NET SAVE AREA 4095            NET SAVE AREA 4095
DC1 OK                  DC1               EQ 1                             1
DC4095 OK               DC4095            EQ 4095                          4095
SW1 OK                  SWITCH               SW1                           1
NSW4095 OK              SWITCH               NSW4095                       1
SW1&TSW666&NSW4095 OK   SWITCHES             SW1&TSW666&NSW4095            111
NSW3000|SW99|TSW666 OK  SWITCHES             NSW3000|SW99|TSW666           001
SW1|..|SW7 OK           SWITCHES             SW1|SW2|SW3|SW4|SW5|SW6|SW7+  1000000001000000
SW10|..|SW7 OK          SWITCHES             SW10|SW2|SW3|SW4|SW5|SW6|SW7+ 0000000001000000
```

Figure 9. Example of a Verification Detail Report

Each Verification Detail Report record contains fields for a description, location, length, condition, expected value, and actual value. The contents of some of these fields vary. Refer to *WSim Script Guide and Reference* for more information about the VERIFY action of IF statements and VERIFY statements in STL.

The following sections discuss the five general categories of IF statements:

- Data tests

- Event tests
- Cursor position tests
- Switch tests
- Counter and Integer tests.

***Verification Detail Report contents for data tests*:** Data tests are tests that specify the location of character data. These represent character data tests. They test the character data at the designated location. For STL, this is the left side of the simple condition. For the WSim Scripting Language, this is on the LOC or the LOCTEXT=(*data*) operand of the IF statement. These locations are shown below:

- LOC=B+ (or B–)
- LOC=C+ (or C–)
- LOC=D+
- LOC=TH+
- LOC=RH+
- LOC=RU+
- LOC=N+ (or N–)
- LOC=U+ (or U–)
- LOC=*s*+
- LOC=(*row*,*col*)
- LOC=N*s*+
- LOCTEXT=(*data*).

For data tests, the Verification Detail Report fields contain the following information:

Heading	**Field Description**
DESCRIPTION	The first 50 characters of the optional description coded on the VERIFY action.
LOCATION	WSim Scripting Language: The value specified on the LOC or LOCTEXT operand of the IF statement. If LOCTEXT=(*data*) was coded, this value will be TEXT. STL: The WSim resource mapping to the left side of the simple condition (the actual text).
LENG	WSim Scripting Language: The length of the expected text. If the AREA and LENG operands were coded on the IF statement, this value will be the value of the LENG operand. If TEXT=(*data*) was coded, this value will be the length of the data specified. If TEXT=*xx* was coded (test under mask), this value will be 1. STL: The length of the actual text.
COND	WSim Scripting Language: The value of the COND operand on the IF statement, indicating the condition used to evaluate the logic test. Possible values are EQ (equal), NE (not equal), GT (greater than), GE (greater than or equal to), LT (less than), and LE (less than or equal to). STL: The relational operator separating the actual and expected expressions. The values displayed here relate to the STL relational operators as follows: EQ and =, NE and ¬, GT and >, GE and >=, LT and <, and LE and <=.

Heading	Field Description
EXPECTED VALUE	WSim Scripting Language: The expected data as coded on the TEXT=(*data*) operand or as specified by the AREA and LENG operands. Only the first 29 characters (15, if HEX=YES is specified on the VERIFY Loglist Utility command) of the expected data will be printed on the report. If the UTBL operand was coded on the IF statement, there will be no expected data printed. In this case, (UTBL) will be printed for the Expected Value. For tests under mask (TEXT=*xx*), (MASK) will be printed following the mask to be tested. STL: The right side of the simple condition. Only the first 29 characters are printed. If you coded UTBLSCAN, this appears as (UTBL). For tests under mask (&= operator), (MASK) will be printed following the mask to be tested.
ACTUAL VALUE	WSim Scripting Language: The actual data found at the location specified on the LOC operand. If the SCAN operand was coded on the IF statement, there will be no actual data printed. In this case, (SCAN) will be printed for the ACTUAL VALUE. STL: The left side of the simple condition. Only the first 29 characters of data are printed.

Notes:

- Nonprintable characters in the expected and actual data will be translated to periods (.) unless the HEX=YES operand is coded on the Loglist Utility VERIFY command. Refer to “VERIFY and NOVERIFY data type selection commands” on page 60 for more information about the Loglist Utility VERIFY command.
- If either side of a comparison (EXPECTED VALUE or ACTUAL VALUE) is null, then (NULL) will be printed on the appropriate side.

***Verification Detail Report contents for event tests*:** Event tests determine whether an event has been posted. For the WSim Scripting Language, event tests are tests that specify the EVENT operand on the IF statement. For STL, event tests are tests that specify the POSTED function.

For event tests, the Verification Detail Report fields contain the following information:

Heading	Field Description
DESCRIPTION	The first 50 characters of the optional description coded on the VERIFY action.
LOCATION	EVENT.
LENG	(NA)—not applicable for event tests.
COND	(NA)—not applicable for event tests.
EXPECTED VALUE	The name of the event being tested.
ACTUAL VALUE	(NA)—not applicable for event tests.

***Verification Detail Report contents for cursor position tests*:** Cursor position tests are tests that specify the CURSOR operand on the IF statement. They test the current position of the cursor on a simulated display panel. These reports are not applicable for STL.

For cursor position tests, the Verification Detail Report fields contain the following information:

Heading	Field Description
DESCRIPTION	The first 50 characters of the optional description coded on the VERIFY action.
LOCATION	CURSOR.
LENG	(NA)—not applicable for cursor position tests.
COND	(NA)—not applicable for cursor position tests.
EXPECTED VALUE	(*row,col*)—the value coded on the CURSOR=(*row,col*) operand of the IF statement.
ACTUAL VALUE	(*row,col*)—the actual location of the cursor when the test was made.

Verification Detail Report contents for switch tests**:** Switch tests test the settings of one or more network, terminal, or device switches. For the WSim Scripting Language, switch tests specify the LOC operand with one or more switches on the IF statement (for example, LOC=NSW1, LOC=TSW20&SW5). For STL, switch tests test a bit variable or function for true or false. This can be either a shared or unshared bit variable.

For switch tests, the Verification Detail Report fields contain the following information:

Heading	Field Description
DESCRIPTION	The first 50 characters of the optional description coded on the VERIFY action.
LOCATION	SWITCH (or SWITCHES if more than one switch was tested).
LENG	(NA)—not applicable for switch tests.
COND	(NA)—not applicable for switch tests.
EXPECTED VALUE	A string of 24 hexadecimal digits representing the switches named on the LOC operand. The first eight digits represent the 4095 device switches, with the leftmost bit representing SW1 and the rightmost bit representing SW4095. The second eight digits represent the terminal switches named on the LOC operand, and the last eight digits represent the network switches named. If more than one switch was specified on the IF statement, (&) or (|) will follow the 24-digit string indicating the operator that was coded to combine the switches on the LOC operand. Refer to WSim Script Guide and Reference for more information about the LOC operand of an IF statement.
ACTUAL VALUE	Another string of 24 hexadecimal digits. This string represents the actual device, terminal, and network switch settings at the time of the test. The format for the actual value is the same as the format for the expected value.

The following two examples show the format for the expected value and actual value fields for switch tests, depending on how the LOC operand of the IF statement was coded:

Example 1:

LOC=SW1 was coded on the IF statement. There were no switches set at the time of the test.

```
EXPECTED VALUE                ACTUAL VALUE
============================= ==========================
80000000 00000000 00000000    00000000 00000000 00000000
```

Example 2:

LOC=SW1|TSW5|TSW6|TSW7 was coded on the IF statement. Switches actually set at the time of the test were SW9, SW11, and NSW22.

```
EXPECTED VALUE                ACTUAL VALUE
============================= ============================
80000000 0E000000 00000000 (| 00A00000 00000000 00000400
```

***Verification Detail Report contents for counter and number tests*:** For the WSim Scripting Language, counter tests are tests that specify the LOC or LOCTEXT operand with a counter on the IF statement. (For example, LOC=DESQ, LOC=NC7, or LOCTEXT=DC2.) They test the value of the counter specified. Number tests are tests that specify the LOCTEXT=*number* operand on the IF statement. For STL, counter tests are tests of an integer variable or integer expression, such as day_of_week > 15.

For counter and number tests, the Verification Detail Report fields contain the following information:

Heading	Field Description
DESCRIPTION	The first 50 characters of the optional description coded on the VERIFY action.
LOCATION	The name of the counter tested (for example, DSEQ). If LOCTEXT=*number* is coded, this value will be NUMBER.
LENG	(NA)—not applicable for switch tests.
COND	WSim Scripting Language: The value of the COND operand on the IF statement, indicating the condition used to evaluate the logic test. Possible values are EQ (equal), NE (not equal), GT (greater than), GE (greater than or equal to), LT (less than), and LE (less than or equal to). STL: The relational operator separating the actual and expected expressions. The values displayed here relate to the STL relational operators as follows: EQ and =, NE and ¬, GT and >, GE and >=, LT and <, and LE and <=.
EXPECTED VALUE	WSim Scripting Language: Either the numeric value specified on the TEXT=*integer* operand of the IF statement, or the numeric value of the counter named on the TEXT=*cntr* operand. STL: The result of evaluating the expected integer expression coded on the right side of the simple expression.
ACTUAL VALUE	WSim Scripting Language: The actual value of the tested counter. STL: The result of evaluating the actual integer expression coded on the left side of the simple expression.

Verification Summary Reports

The second type of report generated by the VERIFY function is the Verification Summary Report. It is generated at the end of a Loglist Utility run. This report gives the total number of VRFY records encountered for each unique description zone as determined by the ZONE=(*x*,*y*) operand and a total count of all VRFY records encountered. Refer to "VERIFY and NOVERIFY data type selection commands" on page 60 for more information about the ZONE operand. Figure 10 shows an example of the Verification Summary Report.

```
               VERIFICATION SUMMARY REPORT

COUNT   DESCRIPTION ZONE       COUNT   DESCRIPTION ZONE
    7   SCREEN-A VERSION-B        12   SCREEN-A VERSION-C
    2   SCREEN-B VERSION-A         4   SCREEN-X VERSION-Y
  100   SCREEN-1 VERSION-2         2   SCREEN-9 VERSION-1
   29   SCREEN-4 VERSION-5        37   SCREEN-2 VERSION-3
    1   SCREEN-7 VERSION-8         1   SCREEN-7 VERSION-9

TOTAL VERIFY RECORDS = 195
```

Figure 10. Example of the Verification Summary Report

Log record header

The header lines at the top of each output page identify the fields in the log record header. The meanings of the formatted fields are as follows:

NETWORK NAME
The name of the WSim network for this message.

APPCLU/TCPIP/VTAMAPPL NAME
The ID and NAME fields are one of the following:
- For a TCP/IP connection, NAME is the value coded in the name field of the TCPIP statement.
- For a VTAM application, NAME is the value coded in (or defaulted to) the name field of the VTAMAPPL statement.
- For a CPI-C transaction program, NAME is the value coded in (or defaulted to) the name field of the APPCLU statement.

DEV/LU/TP NAME
The name of the simulated resource that generated or received this message. If the resource is an SNA logical unit, the current session number is appended to the LU name. If the resource is a CPI-C transaction program (TP), the current TP instance number is appended to the TP name.

START TIME, STOP TIME, READY TIME
These are the message time stamps. For more information about time stamps, refer to Chapter 12, "Understanding message logging," on page 163.

RECORD TYPE
The record type can be one of the following. Refer to Chapter 12, "Understanding message logging," on page 163 for more information about record types.

XMIT Data transmitted by a simulated resource.
Note: These records can be prefixed by % or +. See "Information you can obtain with the Loglist Utility" on page 25 for an explanation of these prefixes.

RECV	Data received by a simulated resource. **Note:** These records can be prefixed by % or +. See “Information you can obtain with the Loglist Utility” on page 25 for an explanation of these prefixes.
INFO	Informational data written as the result of an error or a user request
MARK	Marker record to indicate one minute of elapsed time in the simulation run
MTRC	Message generation trace records for a simulated resource
STRC	STL trace records for a simulated resource
CNSL	Console record containing a WSim operator command or its response
DSPY	3270 and 5250 display or printer buffers
LOG	Data written as a result of the LOG statement or IF statement LOG operand
VRFY	Verification report data
CTRC	CPI-C trace records for a simulated transaction program.

HEADER FLAGS
The two bytes of record type flags and three bytes of modifier flags from the log data set record header.

DATA LENG
The length of the data in the record not including the log header.

TYPE The 1-byte WSim code for the simulated resource. The codes are defined in Chapter 11, “Simulated resource type codes,” on page 161.

MESSAGE DECK
The name of the current message generation deck for the simulated resource.

USER DATA
The 1-byte user field set by the TEXT statement printed in hexadecimal and character formats.

SEQUENCE NUMBER
The number of messages written to the log data set for the simulated resource.

Note: Display images (logged as a result of the LOGDSPLY operand) will be sequenced separately from all other record types for a device.

Log display attribute table header

The header lines of the attribute table for log display records (DSPY) identify bit definitions of the attribute byte. You can get the attribute table when you code the ATTR operand on the DSPLY command. Refer to Figure 12 on page 40 for an example of Loglist Utility output. The following fields are used in the attribute table header:

ROW	The screen image row where this attribute byte is located
COL	The screen image column where this attribute byte is located
U/P	Indicates whether the field is protected or unprotected
A/N	Indicates an alphanumeric or numeric field

HI	Indicates high intensity
SEL	Indicates whether or not the field is selector-pen detectable
NDP	Indicates a nondisplayable, nonprintable, or nondetectable field
MDT	Modified data tag indicator
HLT	Indicates type of highlighting (underline, reverse, or blink)
COLOR	Indicates color of field
CST	Gives the character set used
MF	Indicates a mandatory fill field
ME	Indicates a mandatory enter field
TG	Indicates a trigger field
FO	Indicates type of outlining (left, right, top, or bottom)
SOSI	Indicates whether or not shift out/shift in is enabled for field.

Running the Loglist Utility

The Loglist Utility requires information about how you want to format the output of the log data set. You supply this information by using control commands entered from an input file or the console. The JCL or TSO CLIST you use to run the Loglist Utility provides the following locations:

- Log data set
- Control command input file
- Loglist Utility program
- The printer

The following sections describe more information about the Loglist Utility:

- Coding control commands
- Execution parameters
- Examples of JCL and a TSO CLIST
- Output generated by a sample control command file

Note: If you want to write your own log analysis program, the Loglist Utility can read the log, assemble segmented records, and, as an option, select specific record types for processing. The Loglist Utility then passes these records to the user-written analysis program using the Loglist Utility EXIT facility. For information about user exit routines for the Loglist Utility, see *WSim User Exits*.

Coding the output format

To format output from the Loglist Utility, you can code control commands that tell the utility which records to list. If you issue the utility's RUN command without specifying any other control commands, the Loglist Utility formats every record type for all of the simulated resources.

Table 1 on page 36 shows what causes the various types of log records to be written to the log data set during WSim execution and which Loglist Utility commands can be used to specify the inclusion of these records in the Loglist Utility output.

Table 1. WSim Log record type cross reference

Record Type	WSim Execution	Loglist Utility Command
Console	Any messages written to the WSim operator console, including those written from a script	CNSL
Information	Automatic	INFO
Log	LOG statement and LOG operand on IF statement	LOG
Log Display	LOG statement and LOGDSPLY operand	DSPLY
Marker	Automatic	
Message Data	MLOG=YES	DATA
Message Trace	MSGTRACE=YES	MTRC
STL Trace	STLTRACE=YES	STRC
VRFY	VERIFY operand on IF statement and VERIFY statement in STL	VERIFY
CPI-C Trace	CPITRACE=VERB or CPITRACE=VERBEND or CPITRACE=MSG	CTRC

See Chapter 5, "Specifying loglist control commands," on page 45 for additional information on coding loglist control commands.

Using the WSim/ISPF Interface

You can run the Loglist Utility from the WSim/ISPF Interface. To do this, follow these steps:

1. Invoke the WSim/ISPF Interface main panel from ISPF. The method you use to do this depends on how the WSim/ISPF Interface application is installed at your site. If you are not sure how to do this, see your system programmer for assistance.
2. Select option 7 from the WSim/ISPF Interface main panel and press **Enter**. The Analyze Logged Data panel is displayed.

   Note: You can also type "LOGLIST" on the command line and press **Enter** to display this panel.
3. Fill in the appropriate information on this panel and press **Enter** to run the Loglist Utility.

For more information on the WSim/ISPF Interface, see Chapter 2, "Running WSim with the WSim/ISPF Interface," on page 5.

Using Loglist Utility execution parameters

You can enter the following optional execution parameters in the PARM field for the JCL EXEC statement or on the CALL statement for TSO CLISTS when you run the Loglist Utility.

CONSOLE
Specifies that a write-to-operator-with-reply (WTOR) is issued to the WSim operator console for the input control commands. If you do not specify CONSOLE, the control commands are read from the SYSIN data set.

PRMODE=(LINE|SOSI1|SOSI2)
Indicates the type of system printer support available. When printing DBCS data in 3270 log display records, specifying PRMODE indicates whether the system printer leaves a space when an SO or SI character is printed. DBCS data in console records is printed as-is.

You can specify the following values for PRMODE:

LINE Indicates that the system printer cannot print DBCS data. This is the default value.

SOSI1 Indicates the system printer leaves a space when an SO or SI character is encountered. When you specify PRMODE=SOSI1, extra blank characters are not inserted into print records containing DBCS data delimited by SO and SI characters.

SOSI2 Indicates that the system printer does not leave a space when an SO or SI character is printed. When you specify PRMODE=SOSI2, extra blank characters are inserted into print records containing DBCS data delimited by SO and SI characters. These blanks maintain the character spacing of simulated 3270 devices that display SO and SI characters as blanks.

You must specify PRMODE=SOSI1 or PRMODE=SOSI2 to print DBCS data. If you specify PRMODE=LINE, or do not specify PRMODE, DBCS data in 3270 log display records is formatted as SBCS data. This allows screen images containing DBCS data to be printed on a non-DBCS printer.

PRTLNCNT=*nnn*
The PRTLNCNT parameter specifies the maximum number of lines to be printed on a page of output before ejecting to a new page. The value for *nnn* is an integer from 35 to 255. The default value for *nnn* is 60.

Note: If you want to see the entire screen for display images, increase PRTLNCNT to be at least the number of lines on the screen plus 17. This, however, may cause unusual page breaks or discarded output if you specify more lines than your printer can actually print.

ROUTCDE=(*n***,***n***,...)**
The ROUTCDE parameter specifies the message routing codes to be used in writing Loglist Utility messages to the operator. Each *n* is a system routing code that defines a console destination for every write-to-operator (WTO) and WTOR message written by the Loglist Utility. The value for *n* is an integer from 1 to 16. The default value for the ROUTCDE parameter is 8.

Using JCL

The following JCL statements are required to run the Loglist Utility on MVS.

Statement	Function
LLJOB1 JOB	Initiates the job.
STEP1 EXEC	Specifies the program name.
STEPLIB DD	Defines the data set containing the WSim host processor modules.

Statement	Function
SYSPRINT DD	Defines the output printer. SYSPRINT records may be either fixed or variable length. For fixed length records, logical record lengths of 133 to 256 are accepted. For variable length records, logical record lengths of 137 to 260 are accepted. In either case, the maximum length of noncontrol printed data is 255 bytes. The default is fixed 133 byte length with blocking supported. Record lengths larger than 133 are utilized when PRMODE=SOSI1 or PRMODE=SOSI2 is specified. A larger record length allows additional DBCS data to be printed when the formatted 3270 log display images are printed.
SYSUT1 DD	Defines the log data set input file. The SYSUT1 DD statement contains an optional BLKSIZE parameter defining the maximum block size for the input data. If you specify this value, it should be the same as the BLKSIZE parameter on the LOGDD DD statement from the WSim JCL that created the log data set. If you do not specify a BLKSIZE parameter, the value is taken from the LOGDD data set if: 1. It is on a labelled tape and it is not overridden with the JCL (by way of ALLOC) 2. It is on a disk data set and it is not overridden with the JCL (by way of ALLOC).
SYSIN DD	Defines the control command input file.

The following shows an example of the JCL that you can use to run the Loglist Utility and to log the input data set on tape.

```
//LLJOB1   JOB
//STEP1    EXEC PGM=ITPLL
//STEPLIB  DD   DSN=WSIM.SITPLOAD,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//*             MESSAGE LOGGING INPUT DATA SET ON TAPE
//SYSUT1   DD   UNIT=TAPE,VOL=SER=LOGTAP,LABEL=(,NL),
//              DISP=OLD
//SYSIN    DD   *
     NTWRK TESTNET1,TESTNET2
     RUN

     VTAMAPPL VAPPL1
     TERM WSIMAPP1
     TERM WSIMAPP2

     TIME 093600-094100
     RUN
     END
/*
```

The following example shows the JCL that you can use to log the input data set on disk.

```
//LLJOB2   JOB
//STEP1    EXEC PGM=ITPLL
//STEPLIB  DD   DSN=WSIM.SITPLOAD,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//*             MESSAGE LOGGING INPUT DATA SET ON DISK
//SYSUT1   DD   DSN=WSIM.MSGLOG,DISP=SHR
//SYSIN    DD   *
     NTWRK TESTNET1,TESTNET2
     RUN

     VTAMAPPL VAPPL1
     TERM WSIMAPP1
     TERM WSIMAPP2

     TIME 093600-094100
     RUN
     END
/*
```

The commands in the above examples cause the Loglist Utility to process the log data set twice. The first run will list all records on the data set that are for networks TESTNET1 or TESTNET2. After the log data set is on tape, the tape is rewound before the second set of commands (those following the first RUN command) is processed. In the second run, records between the specified time limits for all networks will be considered. The records for all logical units under VTAMAPPL VAPPL1 will be listed. Only the records for logical units WSIMAPP1 and WSIMAPP2 will be listed for VTAMAPPL VAPPL1. Records for any other logical units will not be listed.

Using a TSO CLIST

The following example shows the TSO CLIST that you can use to run the Loglist Utility when the log data set is on disk.

```
ALLOC DDNAME(SYSPRINT) SYSOUT(A)
ALLOC DDNAME(SYSUT1) DSNAME('WSIM.LOGDATA') SHR
ALLOC DDNAME(SYSIN) DSNAME('USER.LLCMNDS.DATA') SHR
CALL  'WSIM.SITPLOAD(ITPLL)'
FREE DDNAME(SYSPRINT)
FREE DDNAME(SYSUT1)
FREE DDNAME(SYSIN)
```

Understanding sample output

The following pages contain some examples of the output created when you use JCL, a TSO CLIST or the WSim/ISPF Interface to run the Loglist Utility with the input command file shown in Figure 11. Figure 12 on page 40 through Figure 16 on page 42 show the examples.

```
FMTSNA
T D3CNTL1
RUN
NOFMT
TERM D3CNTL1
RUN
TERM D3CNTL1
NOFMT SHORT
RUN
TERM SECLU-1
DSPLY
RUN
TERM SECLU-1
DSPLY ATTR
RUN
```

Figure 11. Command input to the Loglist Utility

```
NETWORK   APPCLU/TCPIP/VTAMAPPL   DEV/LU/TP        START    STOP     READY  RECORD   HEADER      DATA  TYPE  MESSAGE   USER SEQUENCE
 NAME             NAME              NAME           TIME     TIME     TIME    TYPE    FLAGS       LENG         DECK    DATA  NUMBER
WSIMNET1        VA1              LULU2-1       14475538 14475538 14475538  RECV  8000 080000    116   E2   LU2SETUP  00        3
   RECV BIND SESSION REQUEST
       TH   2D0001010000          FID=2   WHOLE SEGMENT  EXPEDITED FLOW  DAF=01   OAF=01   ODAI=0   SEQUENCE=0
       RH   6B8000   REQUEST     SESSION CONTROL      ONLY IN CHAIN   RESPONSE TYPE=DEF1
       RU   31010303 B1903080 008787F8 87000280    00000000 18500000 7E000007 C9E3D7C5   *.........gg8g........&;.=...ITPE*
 00000020   C3C8D600 050009ED 156007D4 E5E2C1D7    D3F16011 D7B7EA26 F189FD0F 08D5C5E3   *CHO......-.MVSAPL1-.P...1i...NET*
 00000040   C14BC1F1 D40E0DF3 D5C5E3C1 4BC9E3D7    C5C3C8D6 2C0A0108 40404040 40404040   *A.A1M..3NETA.ITPECHO....        *
 00000060   2D0908E2 D5E7F3F2 F7F0F2                                                    *...SNX32702                     *
        BIND SESSION   FORMAT 0   TYPE=NON-NEGOTIABLE   FM PROFILE 3     TS PROFILE 3
           PRIMARY PROTOCOLS:   RU CHAINING=MULTIPLE   REQUEST MODE=IMMEDIATE   RESPONSE REQUESTED=DEF OR EXC   END BRACKET SENT
           SECONDARY PROTOCOLS: RU CHAINING=MULTIPLE   REQUEST MODE=IMMEDIATE   RESPONSE REQUESTED=EXCEPTION    END BRACKET NOT SENT
           COMMON PROTOCOLS:    SEGMENTS SUPPORTED     FM HEADERS NOT ALLOWED   BRACKETS RESET BETB      BRACKET TERMINATION RULE 1
                                ALTERNATE CODE SET NOT USED   HALF-DUPLEX FLIP-FLOP    RECOVERY RESPONSIBILITY=PRIMARY
                                CONTENTION WINNER=SECONDARY
           SECONDARY SEND PACING COUNT=NONE        SECONDARY RECEIVE PACING COUNT=07     ADAPTIVE SESSION PACING SUPPORTED
           SECONDARY MAXIMUM RU SEND SIZE=1024     PRIMARY MAXIMUM RU SEND SIZE=3840
           PRIMARY SEND PACING COUNT=07            PRIMARY RECEIVE PACING COUNT=NONE
           LU TYPE 2   DEFAULT SCREEN SIZE=024,080  ALTERNATE SCREEN SIZE=NONE
           PRIMARY LU NAME=ITPECHO                 CRYPTOGRAPHIC FIELD=NONE
           URC=0009ED1560
-------------------------------------------------------------------------------------------------------------------------------------
WSIMNET1        VA1              LULU2-1       14475538 14475538 14475538  XMIT  8000 880000     10   E2   LU2SETUP  00        4
   XMIT BIND SESSION RESPONSE
       TH   2D0001010000          FID=2   WHOLE SEGMENT  EXPEDITED FLOW  DAF=01   OAF=01   ODAI=0   SEQUENCE=0
       RH   EB8000   RESPONSE    SESSION CONTROL      ONLY IN CHAIN   RESPONSE TYPE=DEF1
       RU   31                                                                           *.                               *
-------------------------------------------------------------------------------------------------------------------------------------
WSIMNET1        VA1              LULU2-1       14475538 14475538 14475538  RECV  8000 080000     10   E2   LU2SETUP  00        5
   RECV START DATA TRAFFIC REQUEST
       TH   2D0001010401          FID=2   WHOLE SEGMENT  EXPEDITED FLOW  DAF=01   OAF=01   ODAI=0   SEQUENCE=1,025
       RH   6B8000   REQUEST     SESSION CONTROL      ONLY IN CHAIN   RESPONSE TYPE=DEF1
       RU   A0                                                                           *.                               *
-------------------------------------------------------------------------------------------------------------------------------------
WSIMNET1        VA1              LULU2-1       14475538 14475538 14475538  XMIT  8000 880000     10   E2   LU2SETUP  00        6
   XMIT START DATA TRAFFIC RESPONSE
       TH   2D0001010401          FID=2   WHOLE SEGMENT  EXPEDITED FLOW  DAF=01   OAF=01   ODAI=0   SEQUENCE=1,025
       RH   EB8000   RESPONSE    SESSION CONTROL      ONLY IN CHAIN   RESPONSE TYPE=DEF1
       RU   A0                                                                           *.                               *
-------------------------------------------------------------------------------------------------------------------------------------
WSIMNET1        VA1              LULU2-1       14475538 14475538 14475538  RECV  8000 080000    134   E2   LU2SETUP  00        7
   RECV (DATA) REQUEST
       TH   2C0001010001          FID=2   WHOLE SEGMENT  NORMAL    FLOW  DAF=01   OAF=01   ODAI=0   SEQUENCE=1
       RH   0390C0   REQUEST         FM DATA          ONLY IN CHAIN   RESPONSE TYPE=EXCP   BEGIN-END BRACKET
       RU   F5C7114E 7F1D68E6 C5D3C3D6 D4C540E3    D640C9E3 D7C5C3C8 D64B1D60 40C5D5E3   *5G.+"..WELCOME TO ITPECHO..- ENT*
 00000020   C5D97EC5 C3C8D640 4040C3D3 C5C1D97E    D9C5E2E3 D6D9C540 4040F561 F67EE2E3   *ER=ECHO   CLEAR=RESTORE   5/6=ST*
 00000040   D9C9D5C7 40D9C5D7 C5C1E340 4040F97E    D9C5D7C5 C1E31150 50C5D5E3 C5D940C4   *RING REPEAT   9=REPEAT.&&ENTER D*
 00000060   C1E3C140 E3D640C5 C3C8D640 C2C5D3D6    E67A11D1 5F1D4013 115D7F1D F0         *ATA TO ECHO BELOW:.J¬. ..)".0   *
```

Figure 12. Example of the Loglist Utility output using the FMTSNA command

```
 NETWORK   APPCLU/TCPIP/VTAMAPPL   DEV/LU/TP        START    STOP     READY  RECORD   HEADER      DATA  TYPE  MESSAGE   USER SEQUENCE
  NAME             NAME              NAME           TIME     TIME     TIME    TYPE    FLAGS       LENG         DECK    DATA  NUMBER
 WSIMNET1        VA1              LULU2-1       14475537 14475538 14475537 +XMIT  8000 880020     34   E2   LU2SETUP  00        1
  00000000   2C000001 00010B80 00010681 01E2D5E7    F3F2F7F0 F2F308C9 E3D7C5C3 C8D64000   *...........a.SNX327023.ITPECHO .*
  00000020   0000                                                                        *..                              *
 WSIMNET1        VA1              LULU2-1       14475538 14475538 14475538 +RECV  8000 080020     12   E2   LU2SETUP  00        2
  00000000   2C000100 00018B80 00010681                                                  *...........a                    *
 WSIMNET1        VA1              LULU2-1       14475538 14475538 14475538  RECV  8000 080000    116   E2   LU2SETUP  00        3
  00000000   2D000101 00006B80 00310103 03B19030    80008787 F8870002 80000000 00185000   *.....,..........gg8g........&;*
  00000020   007E0000 07C9E3D7 C5C3C8D6 00050009    ED156007 D4E5E2C1 D7D3F160 11D7B7EA   *.=...ITPECHO......-.MVSAPL1-.P..*
  00000040   26F189FD 0F08D5C5 E3C14BC1 F1D40E0D    F3D5C5E3 C14BC9E3 D7C5C3C8 D62C0A01   *.1i...NETA.A1M..3NETA.ITPECHO...*
  00000060   08404040 40404040 402D0908 E2D5E7F3    F2F7F0F2                              *.        ...SNX32702            *
 WSIMNET1        VA1              LULU2-1       14475538 14475538 14475538  XMIT  8000 880000     10   E2   LU2SETUP  00        4
  00000000   2D000101 0000EB80 0031                                                      *..........                      *
 WSIMNET1        VA1              LULU2-1       14475538 14475538 14475538  RECV  8000 080000     10   E2   LU2SETUP  00        5
  00000000   2D000101 04016B80 00A0                                                      *......,...                      *
 WSIMNET1        VA1              LULU2-1       14475538 14475538 14475538  XMIT  8000 880000     10   E2   LU2SETUP  00        6
  00000000   2D000101 0401EB80 00A0                                                      *..........                      *
 WSIMNET1        VA1              LULU2-1       14475538 14475538 14475538  RECV  8000 080000    134   E2   LU2SETUP  00        7
  00000000   2C000101 00010390 C0F5C711 4E7F1D68    E6C5D3C3 D6D4C540 E3D640C9 E3D7C5C3   *.........5G.+"..WELCOME TO ITPEC*
  00000020   C8D64B1D 6040C5D5 E3C5D97E C5C3C8D6    404040C3 D3C5C1D9 7ED9C5E2 E3D6D9C5   *HO..- ENTER=ECHO   CLEAR=RESTORE*
  00000040   404040F5 61F67EE2 E3D9C9D5 C740D9C5    D7C5C1E3 404040F9 7ED9C5D7 C5C1E311   *   5/6=STRING REPEAT   9=REPEAT.*
  00000060   5050C5D5 E3C5D940 C4C1E3C1 40E3D640    C5C3C8D6 40C2C5D3 D6E67A11 D15F1D40   *&&ENTER; DATA TO ECHO BELOW:.J¬. *
```

Figure 13. Example of the Loglist Utility output using the NOFMT command

```
NETWORK       APPL/TCPIP DEV/LU/TP       START    STOP     READY  RECORD DATA USER
 NAME            NAME      NAME          TIME     TIME     TIME    TYPE  LENG DATA
                                       14472412 0102093  11000000  CNSL    76      Workload Simulator (WSim) Version 1, Release 1.0
                                       14472437 0102093  11000000  CNSL    52      ITP029I INITIALIZATION COMPLETE FOR NETWORK WSIM
                                       14472437 0102093  11000000  CNSL    32      ITP006I NETWORK WSIMNET1 STARTED
                                       14472478 0102093  11000000  CNSL    67      ITP207I DISPLAY MONITOR SESSION STARTED WITH VTA
                                       14474438 0102093  11000000  CNSL    49      ITP137I WSIMNET1 TPCLIENT-00001 - Starting CLIEN
WSIMNET1        VA1      LULU2-1       14475537 14475538 14475537 +XMIT    34 00   ..a.SNX327023.ITPECHO ...
WSIMNET1        VA1      LULU2-1       14475538 14475538 14475538 +RECV    12 00   ..a
WSIMNET1        VA1      LULU2-1       14475538 14475538 14475538  RECV   116 00   .........gg8g........&;.=...ITPECHO......-.MVSAP
WSIMNET1        VA1      LULU2-1       14475538 14475538 14475538  XMIT    10 00   .
WSIMNET1        VA1      LULU2-1       14475538 14475538 14475538  RECV    10 00   .
WSIMNET1        VA1      LULU2-1       14475538 14475538 14475538  XMIT    10 00   .
WSIMNET1        VA1      LULU2-1       14475538 14475538 14475538  RECV   134 00   5G.+"..WELCOME TO ITPECHO..- ENTER=ECHO   CLEAR=
                                       14475638 0102093  11000000  CNSL    50      ITP137I WSIMNET1 LULU2   -00001 - Starting ITPEC
WSIMNET1        VA1      LULU2-1       14475638 14475638 14475638  XMIT    30 00   'J?.J-Hello ITPECHO 1
WSIMNET1        VA1      LULU2-1       14475638 14475638 14475638  RECV    43 00   1C.J-..  .+"..  .  Hello ITPECHO 1
WSIMNET1        VA1      LULU2-1       14475738 14475738 14475738  XMIT    30 00   'J?.J-Hello ITPECHO 2
WSIMNET1        VA1      LULU2-1       14475738 14475738 14475738  RECV    43 00   1C.J-..  .+"..  .  Hello ITPECHO 2
WSIMNET1        VA1      LULU2-1       14475839 14475839 14475839  XMIT    30 00   'J?.J-Hello ITPECHO 3
WSIMNET1        VA1      LULU2-1       14475839 14475839 14475839  RECV    43 00   1C.J-..  .+"..  .  Hello ITPECHO 3
WSIMNET1        VA1      LULU2-1       14475939 14475939 14475939  XMIT    30 00   'J?.J-Hello ITPECHO 4
WSIMNET1        VA1      LULU2-1       14475939 14475939 14475939  RECV    43 00   1C.J-..  .+"..  .  Hello ITPECHO 4
WSIMNET1        VA1      LULU2-1       14480040 14480040 14480040  XMIT    30 00   'J?.J-Hello ITPECHO 5
WSIMNET1        VA1      LULU2-1       14480040 14480040 14480040  RECV    43 00   1C.J-..  .+"..  .  Hello ITPECHO 5
                                       14480042 0102093  11000000  CNSL    39      ITP137I WSIMNET1 TPCLIENT-00001 - Done!
                                       14480140 0102093  11000000  CNSL    39      ITP137I WSIMNET1 LULU2   -00001 - Done!
WSIMNET1        VA1      LULU2-1       14480140 14480140 14480140  XMIT    21 00   'JW.J-logoff
WSIMNET1        VA1      LULU2-1       14480140 14480140 14480140  RECV    34 00   ......-.P...1i...NETA.A1M
WSIMNET1        VA1      LULU2-1       14480140 14480140 14480140 +XMIT    10 00   .
                                       14480437 0102093  11000000  CNSL    45      ITP113I WSIMNET1 DEVSTCP  - Starting MISCSERV
                                       14480692 0102093  11000000  CNSL    44      ITP113I WSIMNET1 DEVTN32E - Starting ITPECHO
                                       14481142 0102093  11000000  CNSL    33      ITP113I WSIMNET1 DEVSTCP  - Done!
                                       14481203 0102093  11000000  CNSL    33      ITP113I WSIMNET1 DEVTN32E - Done!
                                       14550884 0102093  11000000  CNSL     4      zend
                                       14550888 0102093  11000000  CNSL    47      ITP201I DISPLAY MONITOR FACILITY IS CLOSED DOWN
```

Figure 14. Example of the Loglist Utility output using the SHORT operand on the NOFMT command

```
NETWORK   APPCLU/TCPIP/VTAMAPPL   DEV/LU/TP        START    STOP     READY  RECORD   HEADER      DATA  TERM  MESSAGE   USER SEQUENCE
 NAME             NAME              NAME           TIME     TIME     TIME    TYPE    FLAGS       LENG  TYPE    DECK    DATA  NUMBER

USERAPPL        APPL2             SECLU-1        14222660 0096001  11000000  DSPY  0200 080040   2294   E2   SECLU     00         2

            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
 1|                                                                                | 1
 2|                                                                                | 2
 3|       XXXXX  XX XXXXX       XXXX              XXXXX  XX XXXXX       XXX        | 3
 4|       XXXXXX XX XXXXXX     XXXXXX             XXXXXX XX XXXXXX     XXXX        | 4
 5|       XX  XX XX XX  XX     XX  XX             XX  XX XX XX  XX       XX        | 5
 6|       XXXXXX XX XX  XX XXX XX  XX             XXXXXX XX XX  XX XXX   XX        | 6
 7|       XXXXX  XX XX  XX XXX XX  XX             XXXXX  XX XX  XX XXX   XX        | 7
 8|       XX     XX XX  XX     XX  XX             XX     XX XX  XX       XX        | 8
 9|       XX     XX XXXXXX     XXXXXX             XX     XX XXXXXX      XXXX       | 9
10|       XX     XX XXXXX       XXXX              XX     XX XXXXX       XXXX       |10
11|                                                                                |11
12|         ENTER PASSWORD:                         ENTER PASSWORD:                |12
13|                                                                                |13
14|                                                                                |14
15|       XXXXX  XX XXXXX       XXXX              XXXXX  XX XXXXX       XXXX       |15
16|       XXXXXX XX XXXXXX     XXXXXX             XXXXXX XX XXXXXX     XXXXXX      |16
17|       XX  XX XX XX  XX     XX  XX             XX  XX XX XX  XX     XX  XX      |17
18|       XXXXXX XX XX  XX XXX    XX              XXXXXX XX XX  XX XXX    XX       |18
19|       XXXXX  XX XX  XX XXX   XX               XXXXX  XX XX  XX XXX    XX       |19
20|       XX     XX XX  XX      XX                XX     XX XX  XX     XX  XX      |20
21|       XX     XX XXXXXX     XXXXXX             XX     XX XXXXXX     XXXXXX      |21
22|       XX     XX XXXXX      XXXXXX             XX     XX XXXXX       XXXX       |22
23|                                                                                |23
24|            ENTER PASSWORD:                         ENTER PASSWORD:             |24
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
      CURSOR: ROW(  1) COLUMN(  1)   AID: NULL DISPLAY AID           WHEN LOGGED: BEGINNING OF MSG GEN             ACTIVE PID=0
      DIMENSIONS: ( 24, 80)
      PID=0    VIEWPORT SIZE: ( 12, 40)   VIEWPORT LOCATION: (  1,  1) TO ( 12, 40)
      PID=1    VIEWPORT SIZE: ( 12, 40)   VIEWPORT LOCATION: (  1, 41) TO ( 12, 80)
      PID=2    VIEWPORT SIZE: ( 12, 40)   VIEWPORT LOCATION: ( 13,  1) TO ( 24, 40)
      PID=3    VIEWPORT SIZE: ( 12, 40)   VIEWPORT LOCATION: ( 13, 41) TO ( 24, 80)
```

Figure 15. Example of the Loglist Utility output using the DSPLY command

```
NETWORK   APPCLU/TCPIP/VTAMAPPL   DEV/LU/TP        START    STOP     READY RECORD   HEADER      DATA  TERM  MESSAGE   USER SEQUENCE
 NAME             NAME              NAME           TIME     TIME     TIME    TYPE    FLAGS      LENG  TYPE    DECK    DATA  NUMBER

USERAPPL        APPL2             SECLU-1        14222660 0096001  11000000  DSPY  0200 080040   2294   E2   SECLU     00         2

                       A T T R I B U T E                                                   A T T R I B U T E
ROW COL U/P A/N HI SEL NDP MDT HLT COLOR CST  MF ME TG FO   SOSI    ROW COL U/P A/N HI SEL NDP MDT HLT COLOR CST  MF ME TG FO   SOSI
  3   7  P   S                                                        3  47  P   S
  4   7  P   S                                                        4  47  P   S
  5   7  P   S                                                        5  47  P   S
  6   7  P   S                                                        6  47  P   S
  7   7  P   S                                                        7  47  P   S
  8   7  P   S                                                        8  47  P   S
  9   7  P   S                                                        9  47  P   S
 10   7  P   S                                                       10  47  P   S
 12   9  P   S   X  X                                                12  25  U              X
 12  31  P   S                                                       12  49  P   S   X  X
 12  65  U              X                                            12  71  P   S
 15   7  P   S                                                       15  47  P   S
 16   7  P   S                                                       16  47  P   S
 17   7  P   S                                                       17  47  P   S
 18   7  P   S                                                       18  47  P   S
 19   7  P   S                                                       19  47  P   S
 20   7  P   S                                                       20  47  P   S
 21   7  P   S                                                       21  47  P   S
 22   7  P   S                                                       22  47  P   S
 24   9  P   S   X  X                                                24  25  U              X
 24  31  P   S                                                       24  49  P   S   X  X
 24  65  U              X                                            24  71  P   S
NUMBER OF ATTRIBUTES= 44   P=PROTECTED   U=UNPROTECTED   N=NUMERIC   S=AUTOMATIC SKIP
COLOR: B=BLUE  R=RED  P=PINK  G=GREEN  T=TURQUOISE  Y=YELLOW  W=WHITE
```

Figure 16. Example of the Loglist Utility output using the ATTR operand on the DSPLY command

Understanding Loglist Utility return codes

The only return code the Loglist Utility sets is 0, indicating the loglist was formatted as specified.

Chapter 5. Specifying loglist control commands

The following sections provide an explanation of the control commands and operands you can use to operate the Loglist Utility. To help you use this information, they also explain the requirements to enter the coding and the coding conventions for control commands and the categories of commands.

Coding the control commands

You can enter a command in any position of the input record. Operands cannot extend past column 71, and they cannot be continued. Although you can separate a command and its operand by more than one blank space, you must enter at least one blank space between the command and its operand.

You must code all control commands in uppercase when using the control command input file.

You can abbreviate some of the Loglist Utility control commands to a single letter. For example,

```
D dev55
```

is equivalent to

```
DEV dev55
```

Note that the allowable abbreviations for the commands are listed underneath the full command keyword. You can code either the full keyword or the abbreviation.

Understanding control command coding conventions

The following conventions are used in the control command descriptions:

- Capital letters represent values you code directly without change.
- Italics represent parameters for which you must supply a value.
- Brackets, [and], enclose operands or symbols that are either optional or conditional.

  An optional operand is an operand that you may choose to code or omit, independent of other operands. Omitting it may cause a specific default value to be assumed. The default value is always given in the operand description.

  A conditional operand is an operand that you may need to code or omit, depending on how you code (or omit) other operands on the control statement. For each conditional operand, the conditions under which you should code or omit the operand are indicated.
- Braces, { and }, indicate that an operand has a value that you must choose from the stacked items.
- An ellipsis in parentheses, (...), indicates that you may code a sequence of values within parentheses.
- Default values are underlined.

Control command categories

Control commands are in one of the following categories:

- General control
- Record selection

General control commands

General control commands specify general output characteristics and control the execution of the loglist.

The following are general control commands:

END
EXIT
FMTSNA
HEADER
NOFMT
P
RUN
UPCASE
*

Record selection

Record selection control commands determine collectively which records in the WSim log data set being processed are to be included in the loglist output. The record selection category includes these subcategories:

Resource selection control commands
These commands select log records according to the simulated resource for which they are logged.

Primary resource selection control commands
These commands point to a higher level resource such as a LINE or VTAMAPPL which may include one or more lower level resources.

Secondary resource selection control commands
These commands indicate more specific lower level resources to be included or excluded, such as TERM or DEV.

Data type selection control commands
These commands specify the types of log records to be included or excluded from the loglist output.

Overall selection control commands
These commands specify broader criteria for selecting the records beyond resource names and data types.

Table 2 on page 47 shows which control commands belong to each category.

Table 2. Record selection control commands

Overall Selection	Resource Selection	Data Type Selection
MSGTXT NTWRK TIME	Primary Resource Selection APPCLU TCPIP VTAMAPPL Secondary Resource Selection DEV EXDEV EXTERM EXTP TERM TP	CNSL and NOCNSL CTRC and NOCTRC DATA and NODATA DSPLY and NODSPLY INFO and NOINFO LOG and NOLOG MTRC and NOMTRC NOHDR STRC and NOSTRC VERIFY and NOVERIFY

Notes:

- If no selection control commands are entered, all records in the log data set will be included in the output.
- If any resource selection control command is entered, the output will be restricted to records for those resources thus selected (and CNSL commands).
- If any data type selection commands are entered, the output will be restricted to those types included or those types not specifically excluded (except that CNSL records are always listed unless specifically excluded).
- If a mixture of types of record selection commands are entered, only those records that meet all criteria specified will be included in the output (again, CNSL records are always included unless specifically excluded).
- All records, including CNSL records, must fall within the time limits specified to be included in the output.
- If a secondary resource selection command is entered before any primary resource selection command, primary resource selection commands will no longer be valid. Secondary resource selection commands entered after primary resource selection commands bear a hierarchical relationship to the first preceding primary resource selection command.
- Inclusive and exclusive secondary resource selection commands cannot be mixed under the same primary resource selection command or when no primary resource selection command is specified.

APPCLU primary resource selection command

```
APPCLU name
A name
```

The APPCLU command specifies the name of an APPC LU in the simulation run for which the listing will be performed.

name

Function: The *name* operand specifies the name of the APPC LU.

Format: The value of the *name* operand is a 1- to 8-character name that matches the name coded on an APPCLU statement in the WSim network definition.

CNSL and NOCNSL data type selection commands

```
CNSL
```

The CNSL command specifies that console command and console response records are to be printed.

```
NOCNSL
```

The NOCNSL command specifies that console records are not to be printed. Unless NOCNSL is specified, CNSL records will always be printed.

CTRC and NOCTRC data type selection commands

```
CTRC
```

The CTRC command specifies that CPI-C trace records are to be printed. When WSim prints CPI-C trace records, it formats them in a form similar to that used in *SAA Common Programming Interface Communications Reference*. The formatted verbs show all the current parameter data, along with an indication of whether the parameter is input or output.

If you specify the NOFMT control command, the separator lines do not appear in the formatted loglist. If you specify NOFMT SHORT, WSim produces an abbreviated form of the loglist with each CPI-C verb on a single line.

Note: CPI-C trace records will be written to the log data set for a particular transaction program only if CPI-C tracing was requested for the transaction program during the simulation run. CPI-C tracing is requested using the CPITRACE operand of the TP network definition statement. CPITRACE must be specified or defaulted to VERB, VERBEND, or MSG before CPI-C trace records for the transaction program will be written to the log data set. If you specify CPITRACE=NONE, no CPI-C trace data will be in the WSim log. The Loglist Utility will be unable to print CTRC records if no CPI-C trace records are in the log data set. For more information about the CPITRACE operand, refer to *WSim Script Guide and Reference*.

```
NOCTRC
```

The NOCTRC command specifies that CPI-C trace records are not to be printed. This command is required if CPI-C tracing is requested, but CPI-C trace records are not to be printed.

DATA and NODATA data type selection commands

```
DATA
```

The DATA command specifies that transmit and receive data records are to be printed.

```
NODATA
```

The NODATA command specifies that transmit and receive data records are not to be printed.

DEV secondary resource selection command

```
DEV name[-num]
D name[-num]
```

The DEV command specifies the name of the device, logical unit (LU), or transaction program (TP) for which the listing will be performed.

DEV, TERM, and TP commands can be used interchangeably.

If you want to specify individual secondary resources, you must enter a secondary resource selection command, such as DEV, for each individual device, LU, or TP to be listed. If the DEV command does not follow a valid primary resource selection command, the *name* specified will be listed for all primary resources. If the DEV command follows a valid primary resource selection command, the *name* specified will be listed for that primary resource only. If no DEV or other secondary resource selection commands are entered following a primary resource selection command, then all secondary resources for the specified primary resource are listed.

name

Function: The *name* operand specifies the name of the device, logical unit (LU), or transaction program (TP).

Format: The value of the *name* operand is a 1- to 8-character name that matches the label name on a WSim network definition DEV, LU, or TP statement.

num

Function: The *num* operand specifies either a single session number for an LU with multiple session capability or a specific transaction program instance.

Format: For an LU, *num* can be any decimal integer from 1 to 65535. For TP instances, *num* can be any decimal integer from 1 to 99999.

Note: If *num* is not appended to the LU name and multiple sessions exist for the LU, all of the sessions are considered. If *num* is not appended to the TP name and multiple instances exist for the TP, all of the instances are considered.

DSPLY and NODSPLY data type selection commands

```
DSPLY [ATTR]
      [,NONDISP]
      [,PTNS]
```

The DSPLY command specifies that 3270 and 5250 log display records and 3270 printer buffer records are to be formatted and printed.

The log display records for partitions will be formatted to look the same as the operator sees them on the panel. The partition viewports will be displayed together as a single screen image. A map of the presentation space will be displayed following the screen image.

Note: Only those display records (or printer buffer records) that were recorded in the log data set can be printed. An LU1 is not a supported printer buffer record type but an LU3 is a supported printer buffer record type. For the devices mentioned above, you have the option of:

- Not logging the display buffer at all
- Logging the display buffer at the beginning of message generation
- Logging the display buffer at the end of message generation
- Logging the display buffer both at the beginning and end of message generation.

If (and when) display and printer buffer logging takes place is determined by the value of the LOGDSPLY operand. Refer to *WSim Script Guide and Reference* for more information about the LOGDSPLY operand.

ATTR
: **Function:** The ATTR operand causes field and extended field attribute specifications to be listed.

  Note: The ATTR operand is not applicable to 3270 printer buffer formatting.

NONDISP
: **Function:** The NONDISP operand causes nondisplayable fields to be printed. Fields designated as nondisplayable (password fields, for example) will not be printed unless the NONDISP operand is specified.

PTNS
: **Function:** The PTNS operand also allows the partition presentation spaces for each partition to be displayed separately.

```
NODSPLY
```

The NODSPLY command specifies that 3270 and 5250 log display records and 3270 printer buffer records are not to be formatted and printed.

END control command

```
END
```

The END command specifies that the Loglist Utility is to come to a normal completion. No further processing occurs. This command is required. Any commands entered since the last RUN command are ignored.

EXDEV secondary resource selection command

```
EXDEV name[-num]
EXD name[-num]
```

The EXDEV (exclude device) command specifies the name of the device, logical unit (LU), or transaction program (TP) for which the listing will not be performed.

The EXDEV, EXTERM, and EXTP commands can be used interchangeably.

If you want to specify individual secondary resources, you must enter a secondary resource selection command, such as EXDEV, for each individual terminal, device, LU, or TP which you want to exclude from the listing.

If the EXDEV command follows a valid primary resource selection command, the *name* specified will be excluded from the listing for that primary resource only. If the EXDEV command does not follow a valid primary resource selection command, the *name* specified will be excluded from the listing for all primary resources.

name

Function: The *name* operand specifies the name of the terminal, device, logical unit (LU), or transaction program (TP).

Format: The value of the *name* operand is a 1- to 8-character name that matches the label name on a WSim network definition DEV, LU, or TP statement.

num

Function: The *num* operand specifies either a single session number for an LU with multiple session capability or a specific transaction program instance.

Format: For an LU, *num* can be any decimal integer from 1 to 65535. For TP instances, *num* can be any decimal integer from 1 to 99999.

Note: If *num* is not appended to the LU name and multiple sessions exist for the LU, all of the sessions are excluded. If *num* is not appended to the TP name and multiple instances of the TP exist, all instances are excluded.

EXIT control command

```
EXIT [member]
```

The EXIT command specifies the user exit routine to be loaded by the Loglist Utility and to be given control each time a log data set record is read that satisfies the specifications of the other input control commands. Refer to *WSim User Exits* for more information on user exits.

Note: If the EXIT command is omitted or is coded without a member name, the current exit routine will be deleted.

member

Function: The *member* operand specifies the name of the user exit routine.

Format: The value of *member* is a 1- to 8-character name that conforms to standard JCL naming conventions.

EXTERM secondary resource selection command

```
EXTERM name[-num]
EXT name[-num]
```

The EXTERM (exclude terminal) command specifies the name of the device, logical unit (LU), or transaction program (TP) for which the listing will not be performed.

EXDEV, EXTERM, and EXTP commands can be used interchangeably.

If you want to specify individual secondary resources, you must enter a secondary resource selection command, such as EXTERM, for each individual device, LU, or TP that you want to exclude from the listing.

If the EXTERM command follows a valid primary resource selection command, the *name* specified will be excluded from the listing for that primary resource only. If the EXTERM command does not follow a valid primary resource selection command, the *name* specified will be excluded from the listing for all primary resources.

name

Function: The *name* operand specifies the name of the device, logical unit (LU), or transaction program (TP).

Format: The value of the *name* operand is a 1- to 8-character name that matches the name coded on a WSim network definition DEV, LU, or TP statement.

num

Function: The *num* operand specifies either a single session number for an LU with multiple session capability or a specific transaction program instance.

Format: For an LU, *num* can be any decimal integer from 1 to 65535. For TP instances, *num* can be any decimal integer from 1 to 99999.

Note: If *num* is not appended to the LU name and multiple sessions exist for the LU, all of the sessions are excluded. If *num* is not appended to the TP name and multiple instances of the TP exist, all instances are excluded.

EXTP secondary resource selection command

```
EXTP name[-num]
```

The EXTP (exclude transaction program) command specifies the name of the transaction program for which the listing will not be performed.

You must enter an EXTP command for each individual transaction program that you want to exclude from the listing.

EXDEV, EXTERM, and EXTP commands can be used interchangeably.

If the EXTP command follows a valid APPCLU command, the *name* specified will be excluded from the listing for that APPC LU only. If the EXTP command does not follow a valid APPCLU command, the *name* specified will be excluded from the listing for all APPC LUs.

name
: **Function:** The *name* operand specifies the name of the transaction program (TP).

  Format: The value of the *name* operand is a 1- to 8-character name that matches the name coded on a WSim network definition TP statement.

num
: **Function:** The *num* operand specifies a specific transaction program instance.

  Format: The value for *num* can be any decimal integer from 1 to 99999.

  Note: If *num* is not appended to the TP name and multiple instances of the TP exist, all instances are excluded.

FMTSNA and NOFMT control commands

```
FMTSNA
```

The FMTSNA command specifies that special formatting of SNA record headers and Data Link Control (DLC) records is to occur. FMTSNA is the default formatting option for all Loglist Utility runs. Any SNA or DLC records encountered are decoded, and word descriptions of bit meanings and command bytes are printed. All output records (SNA or non-SNA) are separated by a dashed line.

```
NOFMT [SHORT]
```

The NOFMT command specifies that SNA and DLC records are not to be formatted. This command is required if SNA or DLC formatting is not desired. If NOFMT is specified, the dash line is not used to separate records on the output report.

SHORT
: **Function:** The SHORT operand specifies that all records will be printed in abbreviated format on only one output line each, and that SNA records will not be formatted into prose descriptions.

  For SNA terminals, the first 50 bytes of the RU data will be printed.

  For CPI-C transaction programs, the CPI-C trace formatting is abbreviated when NOFMT SHORT is specified.

  For all terminals, any unprintable characters in the data stream will be translated to periods (.).

HEADER control command

```
HEADER data
```

The HEADER command specifies a header line for the formatted log data set output.

data

Function: The *data* operand specifies the header line to be used.

Format: The *data* operand can be up to 27 characters, including blanks and special characters. The *data* operand begins with the first nonblank character after the HEADER command and ends with the character in column 71 or after 27 characters.

Default: If the HEADER command is omitted, the output header defaults to "WSim LOGLIST OUTPUT".

INFO and NOINFO data type selection commands

```
INFO
```

The INFO command specifies that the informational records are to be printed. If INFO is not specified, informational records will be listed for those resources matching other Loglist Utility command specifications.

```
NOINFO
```

The NOINFO command specifies that informational records are not to be printed.

LOG and NOLOG data type selection commands

```
LOG
```

The LOG command specifies that those records with the LOG type are to be printed. These are records written to the log data set as a result of the LOG statement or the LOG operand of the IF statement.

```
NOLOG
```

The NOLOG command specifies that records with the LOG type are not to be printed.

MSGTXT overall selection command

```
MSGTXT name
M name
```

The MSGTXT command specifies the name of a message generation deck (or STL procedure) for which the listing will be performed.

If any MSGTXT commands are entered, records selected will be limited to those containing a MSGTXT name equal to one of those entered or containing no MSGTXT name at all.

name
: **Function:** The *name* operand specifies the name of the message generation deck or STL procedure.

  Format: The value of the *name* operand is a 1- to 8-character name that matches the label name on a WSim message generation or STL MSGTXT statement.

MTRC and NOMTRC data type selection commands

```
MTRC
```

The MTRC command specifies that MTRC records are to be printed. You can use these trace records to follow the steps through message generation. These records help determine if messages are being sent as expected, if the proper IF statements are activated at the time of logic test, when calls and branches are taken, and when EVENTs are signaled, posted, or reset.

Note: You must specify MSGTRACE=YES for a device, LU, or TP before message trace records will be written to the log data set during a simulation run. The Loglist Utility will be unable to print MTRC records if no message trace records are in the log data set. MTRC records reference the sequence numbers from the message generation statements. For more information about the MSGTRACE operand, refer to *WSim Script Guide and Reference*.

```
NOMTRC
```

The NOMTRC command specifies that MTRC records are not to be printed.

NOHDR data type selection command

```
NOHDR
```

The NOHDR command specifies that “debug blocks,” non-I frames, and records that contain SNA headers only are not to be listed. Refer to *Creating WSim Scripts* for information about creating log records for SNA devices.

NTWRK overall selection command

```
NTWRK network,...
N network,...
```

The NTWRK command specifies the network names for which this RUN applies.

If you do not enter the NTWRK command, all networks in the log data set are listed. If you enter more than one NTWRK command before entering a RUN command, the names on the last NTWRK command are used.

network,...
: **Function:** The *network* is the name of the network for which this run applies. You can enter up to five network names separated by commas.

  Format: The value of the *network* operand is a 1- to 8-character name that matches the name coded on a WSim network definition NTWRK statement.

P control command

```
P
```

The P command specifies that console input is to be terminated, and the Loglist Utility is to begin reading input statements from the SYSIN data set. If this command is entered and the SYSIN data set is not open, the Loglist Utility continues requesting input from the console. The P command is ignored if encountered in the SYSIN data stream.

RUN control command

```
RUN
```

The RUN command specifies that all commands have been entered and that processing of the log data set should begin. This command is required. After processing is complete, all variables are reset to their default values before any more commands are interpreted. If RUN is entered and no other commands have been entered, the entire log data set is processed.

STRC and NOSTRC data type selection commands

```
STRC
```

The STRC command specifies that STRC records are to be printed. You can use these trace records to follow the execution of STL programs. Refer to *WSim Script Guide and Reference* for more information about STL programs.

Note: You must specify STLTRACE=YES for a device, LU, or TP before STL trace records will be written to the log data set during a simulation run. The Loglist Utility will be unable to print STRC records if there are not STL trace records in the log data set. STRC records reference the sequence numbers from the STL Translator printed output. For more information about the STLTRACE operand, refer to *WSim Script Guide and Reference*.

```
NOSTRC
```

The NOSTRC command specifies that no STRC records are to be printed.

TCPIP primary resource selection command

```
TCPIP name
```

The TCPIP command specifies the name of a TCP/IP connection in the simulation run for which the listing will be performed.

If any resource selection commands are entered, the listing is limited to the designated resources (which may include implied secondary resources as well). If no resource selection commands are entered, all records for all resources are eligible to be included, provided other types of selection criteria are also met.

name

Function: The *name* operand specifies the name of the TCP/IP connection.

Format: The value of the *name* operand is a 1- to 8-character name that matches the name coded on a TCPIP statement in the WSim network definition.

TERM secondary resource selection command

```
TERM name[-num]
T name[-num]
```

The TERM command specifies the name of the device, logical unit (LU), or transaction program (TP) for which the listing will be performed.

DEV, TERM, and TP commands can be used interchangeably.

If you want to specify individual secondary resources, you must enter a secondary resource selection command, such as TERM, for each individual device, LU, or TP to be listed. If the TERM command does not follow a valid primary resource selection command, the *name* specified will be listed for all primary resources. If the TERM command follows a valid primary resource selection command, the *name* specified will be listed for that primary resource only. If no TERM or other secondary resource selection commands are entered following a primary resource selection command, then all secondary resources for the specified primary resource are listed.

name

Function: The *name* operand specifies the name of the device, logical unit (LU), or transaction program (TP).

Format: The value of the *name* operand is a 1- to 8-character name that matches the name coded on a WSim network definition DEV, LU, or TP statement.

num

Function: The *num* operand specifies either a single session number for an LU with multiple session capability or a specific transaction program instance.

Note: If *num* is not appended to the LU name and multiple sessions exist for the LU, all of the sessions are considered. If *num* is not appended to the TP name and multiple instances of the TP exist, all of the instances are considered.

Format: For an LU, *num* can be any decimal integer from 1 to 65535. For TP instances, *num* can be any decimal integer from 1 to 99999.

TIME overall selection command

```
TIME {ALL}
     {x-y}

     wherex can be hhmmss, hhmm, or START
     and y can be hhmmss, hhmm, or END
```

The TIME command specifies the time limits of a simulation run for which the log listing will be performed. The first field indicates the time at which the analysis is to begin. The second field indicates the last second to be processed in the run.

Times are 24-hour clock times (000000-235959). A time interval including midnight is valid, such as TIME 235000-004500. The log data set must contain records between a time that you specify and the beginning of the next hour.

If you do not enter the TIME command, the entire log data set is processed. If you enter two TIME commands, the limits from the last one entered are used. If the last TIME command entered is invalid, TIME ALL is assumed.

Note: Except when using TIME ALL, any format of the first operand field may be paired with any format of the second operand field (for example, TIME 183000-END).

ALL

Function: The ALL operand specifies that the entire log data set is to be analyzed.

hhmmss

Function: The *hhmmss*operand indicates the hours, minutes, and seconds that limit the processing. If you enter this time as*hhmm*, *ss*defaults to *00***.**

START

Function: The START operand indicates that analysis is to begin with the first record on the log data set.

END

Function: The END operand indicates that analysis is to end with the last record in the log data set.

TP secondary resource selection command

```
TP name[-num]
```

The TP command specifies the name of the CPI-C transaction program instance for which the listing will be performed.

DEV, TERM, and TP commands can be used interchangeably.

If you want to specify individual secondary resources, you must enter a secondary resource selection command, such as TP, for each individual transaction program to be listed. If the TP command follows a valid primary resource selection command, the TP instance will be formatted for the primary resource only. If the TP command does not follow a valid primary resource selection command, the *name* specified will be listed for all primary resources. If no TP or other secondary resource selection commands are entered following a primary resource selection command, then all secondary resources for the specified primary resource are listed.

name

Function: The *name* operand specifies the name of the transaction program.

Format: The value of the *name* operand is a 1- to 8-character name that matches the name coded on a WSim network definition TP statement.

num

Function: The *num* operand specifies a single transaction program instance.

Note: If *num* is not appended to the TP name and multiple instances exist for the TP, all of the instances are considered.

Format: The value for *num* can be any decimal integer from 1 to 99999.

UPCASE control command

```
UPCASE
```

The UPCASE command specifies that when the EBCDIC translation of hexadecimal data or formatted log display records is printed, all lowercase characters are translated to uppercase characters.

VERIFY and NOVERIFY data type selection commands

```
VERIFY [ZONE=(x,y)]
      [,HEX={YES|NO}]
```

The VERIFY command causes VRFY log records to be formatted into Verification Detail Reports and also causes the Verification Summary Report to be printed at the end of the Loglist Utility run.

ZONE=(*x*,*y*)
Function: The ZONE operand specifies the portion of the description logged on the VRFY record to be used as a key for accumulating verification statistics for the Verification Summary Report, where *x* specifies the location of the beginning of the zone within each description, and *y* specifies the length of the zone. The *x* and *y* values are integers from 1 to 50. The sum of x and y must be less than or equal to 51.

Default: ZONE=(1,50).

HEX={YES|NO}
Function: The HEX operand specifies whether the expected and actual values for text on Verification Detail Reports should be printed in hexadecimal format.

Format: The HEX operand can have the following values:

YES	Causes the text to be printed in hexadecimal format.
NO	Causes the text to be printed in EBCDIC notation.

Default: NO.

Code HEX=YES to verify locations in a data stream containing nonprintable characters (for example, LOC=TH+, RH+). Code HEX=NO (the default) to verify printable text. Nonprintable characters are translated into periods (.) when HEX=NO.

When HEX=NO, a maximum of 29 characters of the expected and actual text will be printed on the Verification Detail Reports. When HEX=YES, a maximum of 15 characters are translated into hexadecimal format and printed.

```
NOVERIFY
```

The NOVERIFY command inhibits the formatting of VRFY log records into Verification Detail Reports and inhibits the printing of a Verification Summary Report at the end of the Loglist Utility run.

VTAMAPPL primary resource selection command

```
VTAMAPPL name
V name
```

The VTAMAPPL command specifies the name of a VTAMAPPL in the simulation run for which the listing will be performed.

If any resource selection commands are entered, the listing is limited to the designated resources (which may include implied secondary resources as well). If no resource selection commands are entered, all records for all resources are eligible to be included, provided other types of selection criteria are also met.

name
: **Function:** The *name* operand specifies the name of the VTAMAPPL.

  Format: The value of the *name* operand is a 1- to 8-character name that matches the name coded on a VTAMAPPL statement in the WSim network definition.

* control command

```
* [data]
```

The * command specifies a comment. This command can contain any *data* following the asterisk. The command is listed with the other input commands before the output reports, but it is ignored during Loglist Utility processing.

data
: **Function:** The *data* operand can contain any data.

Chapter 6. Using the Log Compare Utility to compare log data sets

The Log Compare Utility enables you to compare the log data set records of panels displayed on a simulated 3270 terminal during two WSim simulations. If the records from the first and second simulation differ, the Log Compare Utility reports the first difference and optionally displays the two screens.

The Log Compare Utility is especially useful when comparing multiple panels. You can use this utility after you alter an application to see if the information on the panels changed in ways you did not anticipate. For example, if your automatic teller machine normally presents a panel showing the user's balance, you may want to change the panel to display the last three transactions as well. After you alter the panel, you can use the Log Compare Utility to tell you exactly what additional information appeared.

Using the Log Compare Utility's control commands, you select the panels to be compared and the fields to be compared on each panel. In addition, you can obtain a set of reports that summarize the differences found between the two log data sets. After comparing the records, the Log Compare Utility lists the commands you entered and the operand values that were in effect while the utility was running.

The sections in this chapter present information about the following:

- DSPY records—what they are and how the Log Compare Utility uses them
- How to use control commands to compare panels
- How to synchronize two log data sets
- The listings and reports the Log Compare Utility provides
- How to run the Log Compare Utility, including:
  - Running the Log Compare Utility from the WSim/ISPF Interface
  - Execution parameters to use with JCL or a TSO CLIST
  - Examples of JCL and a TSO CLIST for running the Log Compare Utility
  - Sample output from an example input command file
  - Return codes after running the Log Compare Utility.

Understanding DSPY records

The Log Compare Utility compares 3270 DSPY records. A 3270 DSPY record is a record type WSim stores in the log data set. The string of information in the DSPY record describes a panel used during a WSim simulation. You can think of a DSPY record as a snapshot of a panel WSim sees as it communicates with the system under test. Each DSPY record in a log data set represents a separate snapshot taken at a different time during the run. The Log Compare Utility uses the panel information to make the comparison.

Figure 17 on page 64 shows the output of a DSPY record as it appears when formatted by the Loglist Utility. For information about the Loglist Utility, see Chapter 4, “Using the Loglist Utility to format the log data set,” on page 25. The DSPY record appears in the log data set output with a header and a panel. Notice that the header identifies the RECORD TYPE as DSPY.

```
NETWORK   APPCLU/TCPIP/VTAMAPPL  DEV/LU/TP         START    STOP     READY  RECORD   HEADER      DATA  TYPE  MESSAGE   USER SEQUENCE
 NAME            NAME              NAME           TIME     TIME     TIME    TYPE     FLAGS       LENG         DECK    DATA  NUMBER
USERAPPL       APPL2           SECLU-1       14222660 0096001  11000000  DSPY  0200 080040   2294   E2   SECLU     00         2
             1         2         3         4         5         6         7         8
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
    --------------------------------------------------------------------------------
  1| ------------------------  ALLOCATE NEW DATA SET  -----------------------------| 1
  2| COMMAND ===>                                                                   | 2
  3|                                                                                | 3
  4| DATA SET NAME: WSIM.SAMPLE.DATASET                                             | 4
  5|                                                                                | 5
  6|    VOLUME SERIAL      ===> USERPK      (Blank for authorized default volume) * | 6
  7|    GENERIC UNIT       ===>             (Generic group name or unit address) *  | 7
  8|    SPACE UNITS        ===> BLOCK       (BLKS, TRKS, or CYLS)                   | 8
  9|    PRIMARY QUANTITY   ===> 540         (In above units)                        | 9
 10|    SECONDARY QUANTITY ===> 250         (In above units)                        | 10
 11|    DIRECTORY BLOCKS   ===> 10          (Zero for sequential data set)          | 11
 12|    RECORD FORMAT      ===> U                                                   | 12
 13|    RECORD LENGTH      ===> 0                                                   | 13
 14|    BLOCK SIZE         ===> 6144                                                | 14
 15|    EXPIRATION DATE    ===>             (YY/MM/DD                               | 15
 16|                                         YY.DDD in Julian form                  | 16
 17|                                         DDDD for retention period in days      | 17
 18|                                         or blank)                              | 18
 19|                                                                                | 19
 20|    ( * Only one of these fields can be specified)                              | 20
 21|                                                                                | 21
 22|                                                                                | 22
 23|                                                                                | 23
 24|                                                                                | 24
    --------------------------------------------------------------------------------
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
             1         2         3         4         5         6         7         8
      CURSOR: ROW(  7) COLUMN( 29)   AID: NULL DISPLAY AID           WHEN LOGGED: BEGINNING OF MSG GEN
      DIMENSIONS: ( 24, 80)
```

Figure 17. Example of a DSPY record as formatted by the Loglist Utility

Each DSPY record can provide the following information:

- The size of the panel
- The data present on the panel
- 3270 field attributes
- 3270 character attributes
- Data about the network resources associated with the device that logged the DSPY record
- 3270 cursor position.

You specify that WSim creates DSPY records during a simulation by coding the LOGDSPLY operand on the NTWRK statement or lower-level device statements in the network definition. After the simulation run, you can run the Log Compare Utility to compare the DSPY records from two different simulations.

For more information about logging DSPY records with the LOGDSPLY operand and the values you can use on the operand, refer to *WSim Script Guide and Reference*. Refer to *WSim User Exits* for additional information about DSPY record formats.

Comparing DSPY records using the Log Compare Utility

When you use the Log Compare Utility to compare two log data sets for a network, the first log data set, called the MASTER log data set, is the reference standard against which the second log data set, called the TEST log data set, is compared. In the reports produced by the Log Compare Utility, you will see the first log data set identified as MASTER and the second as TEST.

The Log Compare Utility compares records for each device in your simulated network separately. By using the control commands, you can select particular lines, terminals, and devices to compare, or you can make comparisons for all devices in your network. You can also select particular records and record fields to compare. Records for each device are numbered sequentially in the log data sets. These sequential numbers appear in the reports produced by the Log Compare Utility.

To run the Log Compare Utility, follow these steps:

1. Run WSim and log the data to the MASTER log data set.
2. Make changes to the system you are testing, for example:
   - Information Management System/Virtual Storage (IMS/VS), Customer Information Control System/Virtual Storage (CICS/VS), DB2® version updates, fixes, modifications, or fine tuning
   - Application program updates, fixes, or modifications.
3. Test your updated system by running WSim and logging the data to the TEST log data set.

   Note: Be sure to store your MASTER and TEST log data sets with different names.
4. Run the Log Compare Utility.
5. Verify that the system remains unchanged or evaluate the changes by studying the output reports. For information about the reports, refer to “Information you can obtain with the Log Compare Utility” on page 72. For examples of the reports, refer to “Understanding sample output” on page 79.

Identifying differences in DSPY records

The Log Compare Utility identifies each comparison as “Equal” or “Not Equal” with a message stating a reason if there is a difference. Two DSPY records are equal when the two screen images are:

- Completely identical in every respect

  The data in each field, as well as all other characteristics, such as location, attributes of the data, and so forth, is identical.
- Slightly different, but you used commands to tell the Log Compare Utility to ignore the differences.

  For example, you can mask a date field from being compared on two different log data sets by using the MASK command. Also, you can exclude records from comparison by identifying a unique string of data in them with the EXCLUDE command.

Two DSPY records are not equal for any one of many reasons, including when the two screen images display:

- Differences in data
- The same data, but not in the same location
- Different field or character attributes

  When you specify the ATTRIBUTE command, the Log Compare Utility identifies attribute as well as data differences.
- Different screen image sizes
- Different cursor positions
- Different screen images because one was logged at the beginning of message generation for a panel and the other was logged after completing message generation for the panel.

  See *Creating WSim Scripts* for an explanation of the message generation process.

When the Log Compare Utility finds a difference between panels, it reports the difference in the Compare Report, the Differences Report, and the Summary

Report. Refer to “Information you can obtain with the Log Compare Utility” on page 72 for information about the reports and “Understanding sample output” on page 79 for examples of reports.

Comparing DSPY records in networks with multiple devices

The Log Compare Utility compares DSPY records (in the MASTER and TEST log data sets) with different or identical device names. Panels for each device are numbered sequentially for the individual device, so if each has five panels, both devices' panels are numbered 0 through 4. If the MASTER and TEST log data sets contain eight different devices in the network for the simulation run, the utility compares each device for which you request comparisons separately.

The Log Compare Utility stops comparing panels for any particular terminal when the total number of panels that have a difference exceeds the threshold amount specified on the ERRCOUNT ABORT=*integer* command. However, the Log Compare Utility continues to compare panels for other terminals. Often, a large number of panels are not equal because the two log data sets are not synchronized when the Log Compare Utility compares them. Refer to “Synchronizing two log data sets” on page 70 for information about the ERRCOUNT command and synchronizing the log data sets.

For more information about comparing multiple devices in log data sets, refer to Chapter 7, “Specifying Log Compare Utility control commands,” on page 91.

Controlling what is compared

You can control how the Log Compare Utility compares panels by using control commands. Control commands manage the selection of records and how records are processed when they are compared. The Log Compare Utility uses two types of commands:
- Selection
- Process.

Understanding selection commands

Selection commands limit the scope of the comparing process and make it more manageable or meaningful. By using selection commands, you identify only the DSPY records you want to compare and allow the Log Compare Utility to ignore the rest.

In addition, you can use selection commands like DEV, LU, or TERM to identify DSPY records for particular devices for which you want to compare simulation runs. For example, although your MASTER and TEST log data sets may each contain DSPY records for DEV1 through DEV100, you may want to compare the DSPY records for only DEV49. In this case, you can code the DEV command as follows:

```
DEV DEV49
```

You may also want to compare the DSPY records for MASTER DEV50 to TEST DEV51. In this case, you can code the DEV command as follows:

```
DEV DEV50,DEV51
```

The Log Compare Utility compares only the MASTER and TEST DSPY records requested. If you code several DEV, LU, or TERM selection commands, the Log Compare Utility compares the MASTER DSPY records for each device to its associated TEST DSPY records.

When you code the VTAMAPPL, TCPIP and NTWRK selection commands, the Log Compare Utility compares a range of network resources. For example, a VTAMAPPL may have several logical units with it, so that when you code:

```
VTAMAPPL VAPPL1
```

the Log Compare Utility compares all logical units associated with VAPPL1. With this coding, the Log Compare Utility extracts only those DSPY records that have VAPPL1, and then compares those records.

If you want to compare the DSPY records for MASTER VAPPL1 to TEST VAPPL2, you can code the VTAMAPPL command as follows:

```
VTAMAPPL VAPPL1,VAPPL2
```

This causes the Log Compare Utility to extract only those DSPY records from the MASTER log data set having VTAMAPPL VAPPL1, and to extract only those DSPY records from the TEST log data set having VTAMAPPL VAPPL2. The Log Compare Utility will then compare those records.

If no VTAMAPPL is specified, the Log Compare Utility extracts those DSPY records matching the logical units specified regardless of the VTAMAPPL.

The MSGTXT selection command provides another way to limit the records that the Log Compare Utility compares. When you specify the MSGTXT command, the Log Compare Utility considers only the DSPY records that were logged while WSim was processing the named message generation deck. Here again, more than one device can be represented in the resulting DSPY records, so the utility compares the MASTER DSPY records that fit the MSGTXT criteria against the TEST DSPY records that fit the MSGTXT criteria for the individual devices being compared.

The EXCLUDE and SELECT commands limit what is compared either by excluding certain records from consideration or by specifically selecting them for consideration. When you define a string of data on the EXCLUDE command, the Log Compare Utility bypasses any DSPY records with a matching string of data. For example, when an application under test has several levels of menus that you do not want to compare, you can define a data string of “menu” on EXCLUDE. Then the Log Compare Utility would exclude any DSPY records containing “menu” from the comparing process. When you are interested only in comparing a single panel, you can code a single SELECT command to define the panel. Then the Log Compare Utility compares only that panel during the comparing process.

The Log Compare Utility also excludes DSPY records produced as a result of the CLEAR command simulating use of the Clear key at a terminal.

The START command defines the panel that is the starting point for the comparing process. For example, the application under test can have several panels of logon and password information that occur before the first panel you are interested in comparing. When you identify that panel by defining a unique data string on the START command, the Log Compare Utility searches for that panel in both the MASTER and TEST log data sets. When the utility locates that panel, it begins the comparing process.

The SYNCPOINT command enables you to define the DSPY record that the utility uses to synchronize the comparison of two log data sets. The ERRCOUNT command enables you to abort or synchronize the comparison after a specific number of mismatches. With the ABORT operand on the ERRCOUNT command, you can specify the number of differences that can be detected for a device before the utility ends the comparing process. When you code the SYNCPOINT operand on the ERRCOUNT command, you specify the number of differences that can be detected for a device before the utility attempts to synchronize the log data sets.

Note: The compare process for devices is completely separate. If you have two devices in your MASTER and TEST log data sets that are not being compared to each other, the Log Compare Utility can abort processing for one device without aborting processing for the other device.

For additional information about synchronization, refer to “Synchronizing two log data sets” on page 70. Refer to “Specifying synchronization with selection commands” on page 70 for more information about using the SYNCPOINT and ERRCOUNT commands.

Understanding process commands

Process commands tell the utility how to process the selected records by specifying the following information:

- What fields to compare
- What fields not to compare
- When to compare the fields
- The reports that the utility is to generate following the run.

When you code the ATTRIBUTE and CHARATTR commands, for example, the Log Compare Utility includes any 3270 data stream attributes in the comparing process. The ATTRIBUTE command specifies that field attributes are to be considered; the CHARATTR command specifies that character attributes are to be considered. Without the ATTRIBUTE command, a field that appears in blinking red would be considered identical to one that appears in reverse video blue, assuming that the contents of the field are also identical. With the ATTRIBUTE command, however, the utility looks at the attributes that define the field and compares them as well. Because blinking red is not equal to reverse video blue, the utility reports that a difference exists.

The CURSOR command specifies that the Log Compare Utility compares the position of the cursor on two records. A difference in cursor positions is considered to be a mismatch.

The CHECKONLY and MASK commands limit the fields that the Log Compare Utility compares on a panel. The CHECKONLY command specifies that the Log Compare Utility consider only certain fields during the comparing process; the MASK command specifies that the utility not consider certain fields during the comparing process. These commands have operands that specify the location of specific records and an operand that defines the area of the panel to be considered during the comparing process:

- The DATA, LOC, and SCAN operands specify the location of data to use to identify a specific record when coded on either the CHECKONLY or MASK command.
- The CHECKLOC operand defines an area of the panel to be considered when you code the CHECKONLY command.

- The MASKLOC operand defines an area of the panel to be ignored when you code the MASK command.

The following examples clarify how you can use these operands and commands to control the comparing process:

CHECKONLY
: If you have DSPY records from two different simulations with many differences between the panels, you may only be interested in the differences that appear between specific fields, for example, the price field and the part number field. By coding a CHECKONLY command with two CHECKLOC operands, you can specify that the utility compares only those two fields and ignores all other fields. During the comparing process, the Log Compare Utility selects the DSPY records defined by the selection commands, and then uses the DATA, LOC, and SCAN operands to identify the record defined by the CHECKONLY command. Then, using the two CHECKLOC operands that you coded, the utility looks at only the price field and the part number field to complete the comparing process.

MASK
: If you have DSPY records that you think are identical except for a few fields, you can code the MASK command to compare every field except those that may have changed. For example, if you have two panels logged within a few hours of one another, the only difference between them might be the field that records the time that the simulation took place. By defining that field with the MASK command, the Log Compare Utility ignores that field during the comparing process. If the rest of the panel is identical, the utility considers the entire panel identical, despite the difference that appears between the time fields.

Table 3 and Table 4 group similar Log Compare Utility commands to help you find the command you need. Commands are grouped to show the levels under which the compare is controlled. These tables list commands from top to bottom, showing from a wide scope of control to a detailed, narrow scope of control.

Table 3. Log Compare Utility selection commands

Command Type	Command	Description
Resource-Level Selection Commands	NTWRK VTAMAPPL, TCPIP DEV, LU, TERM MSGTXT	These commands are used to select resources to compare.
Screen-Level Selection Commands	START EXCLUDE, SELECT	These commands are used to specify which display records to compare.
Screen-Level Error-Recovery Commands	ERRCOUNT, SYNCPOINT	These commands select and discard display records in error recovery situations.

Table 4. Log Compare Utility process commands

Command Type	Command	Description
Partial Screen Compare Commands	CHECKONLY MASK	These commands are used to limit the display record comparison to portions of the record.

Table 4. Log Compare Utility process commands (continued)

Command Type	Command	Description
Compare Commands	ATTRIBUTE, CHARATTR, CURSOR, UPPERCASE	These commands control how the display records are to be compared.
General Control Commands	END, HEADER, P, REPORT, RUN	These commands control the Log Compare Utility and its output.

Synchronizing two log data sets

When the Log Compare Utility detects successive data differences between DSPY records, you can recover and continue to compare the log data sets from a known point.

Data differences may occur between panels because the utility attempted to compare two completely different panels. For example, WSim may log a quantity-on-hand panel as the fifth panel during a simulation performed on Monday. If you then add a new security panel on Tuesday, another simulation performed on Wednesday would log the quantity-on-hand panel as the sixth panel. When the Log Compare Utility compares the fifth panel from Monday's simulation with the fifth panel from Wednesday's simulation, they are entirely different panels.

Although adding a single panel may not seem to be a crucial change, the DSPY records in each pair of panels that follows will fail to compare appropriately. In the preceding example, the panels logged to the MASTER log data set from Monday's simulation and the TEST log data set from Wednesday's simulation are out of sequence by one panel from the point at which you added the new panel. Without the ability to synchronize the two log data sets, running the Log Compare Utility would show significant differences between the two tests, when, in fact, the only difference is that you added one panel.

To recover synchronization when the utility detects a difference in data fields or panels, you establish a “known point” from which the comparing process can be restarted. For example, the utility can return to a primary option menu and then begin comparing panels again. The primary option menu would act as a known point for the Log Compare Utility to restart the comparing process after it became out of sequence.

In effect, the Log Compare Utility says, “I've run into a series of consecutive mismatches, so I'm going to skip ahead to the known point, the primary option menu, and start comparing again.” If the new panel was the only difference between the two tests, then every panel following the known point compares successfully.

Specifying synchronization with selection commands

You can use the SYNCPOINT and ERRCOUNT selection commands to tell the utility when and how to synchronize the log data sets.

The SYNCPOINT operand on the ERRCOUNT command tells the Log Compare Utility how many consecutive mismatches can occur before the comparison is to be considered out of synchronization. After reaching that number of mismatches, the process stops until the utility finds the record identified by a SYNCPOINT command in both log data sets. The utility first searches the MASTER log data set, record by record, for a DSPY record that matches the SYNCPOINT definition. When the utility finds a match, it searches the TEST log data set, record by record, for a DSPY record that matches the same SYNCPOINT definition. The comparing process then resumes from that point.

Note: You can code more than one SYNCPOINT command for each run. When you code more than one, the first one that produces a match is used.

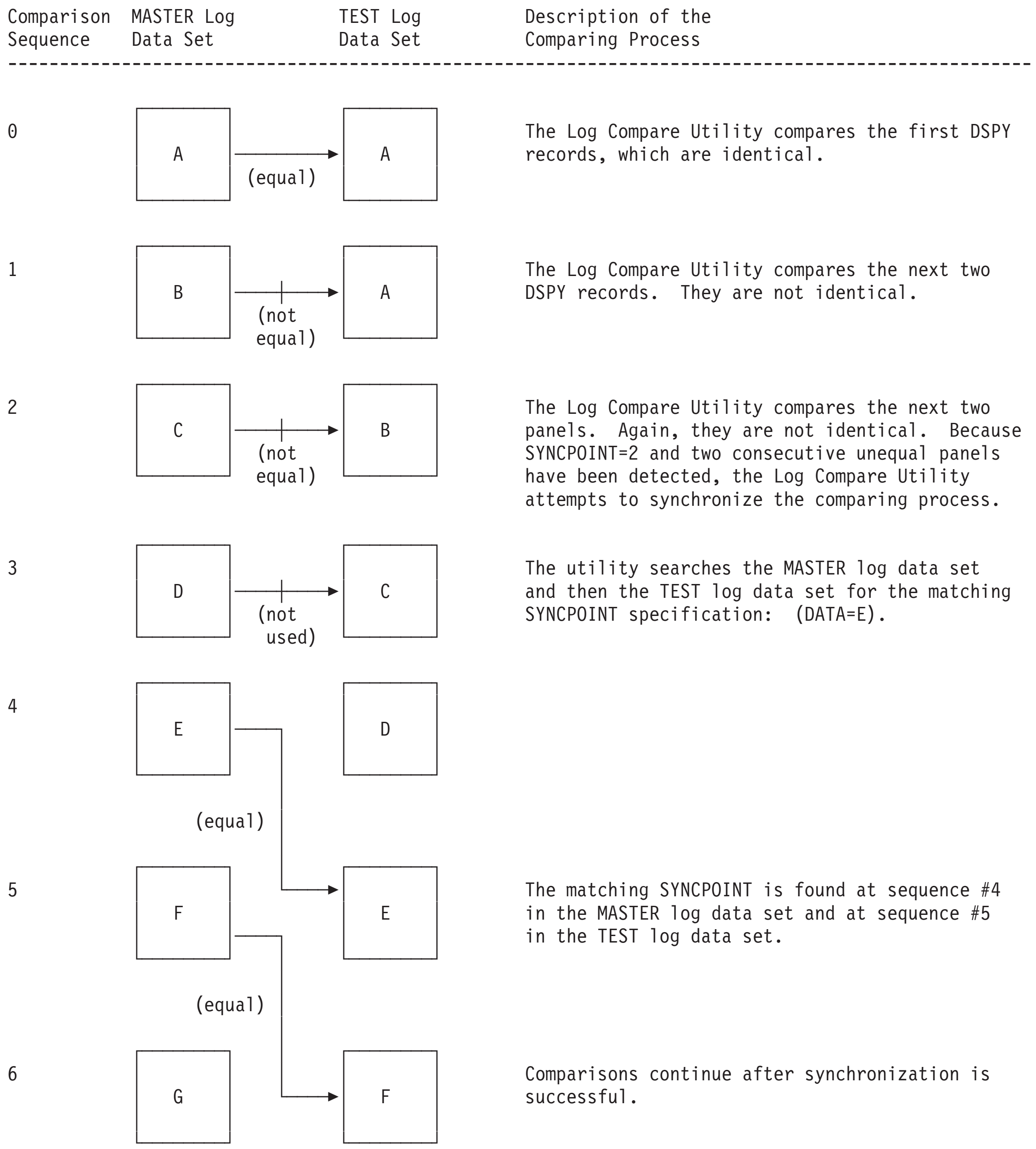
For more information about the ERRCOUNT and SYNCPOINT commands, refer to "Selection commands" on page 92.

Example of log data set synchronization

Figure 18 on page 72 shows how WSim recovers when successive differences are detected between data fields. In this example, each panel contains a field displaying a single letter that identifies the panel. The following selection commands and operands were specified to run the Log Compare Utility:

```
ERRCOUNT  SYNCPOINT=2
SYNCPOINT DATA=(E),
          LOC=(1,1),
          SCAN=YES
```

The operands specify that two consecutive mismatches must occur before the comparison is out of synchronization. The Log Compare Utility then attempts to get back into synchronization by searching DSPY records. The utility starts at location (1,1) and scans to the end of each record for the data E. When the Log Compare Utility finds both a MASTER panel and a TEST panel with E, it begins comparing panels again from this point of synchronization.



```
Comparison  MASTER Log          TEST Log          Description of the
Sequence    Data Set            Data Set          Comparing Process
-----------------------------------------------------------------------------------------------

0                                                 The Log Compare Utility compares the first DSPY
                                                  records, which are identical.

1                                                 The Log Compare Utility compares the next two
                                                  DSPY records.  They are not identical.

2                                                 The Log Compare Utility compares the next two
                                                  panels.  Again, they are not identical.  Because
                                                  SYNCPOINT=2 and two consecutive unequal panels
                                                  have been detected, the Log Compare Utility
                                                  attempts to synchronize the comparing process.

3                                                 The utility searches the MASTER log data set
                                                  and then the TEST log data set for the matching
                                                  SYNCPOINT specification:  (DATA=E).

4

5                                                 The matching SYNCPOINT is found at sequence #4
                                                  in the MASTER log data set and at sequence #5
                                                  in the TEST log data set.

6                                                 Comparisons continue after synchronization is
                                                  successful.
```

Figure 18. The synchronizing process with panel E specified as the SYNCPOINT

Information you can obtain with the Log Compare Utility

With the Log Compare Utility, you can obtain several reports that describe the differences detected between the MASTER and TEST DSPY records:

- Active Command List
- Complete Records List
- Compare List
- Differences Report

- Summary Report.

The Log Compare Utility prints the Active Command List automatically following each run. You control printing of all other reports except the Active Command List by using the REPORT command.

- If you issue the REPORT command without any operands, you will obtain an Active Command List, a Differences Report, and a Summary Report by default.
- If you code the RECORDS, COMPARES, DIFFERENCES, or SUMMARY operands on the REPORT command, the report named by the operand will be printed.
- If you do not specify the REPORT command, you will obtain an Active Command List, a Differences Report, and a Summary Report by default.

The following sections provide detailed information about each report. To see examples of each of the reports, refer to "Understanding sample output" on page 79.

Active Command List

The Log Compare Utility prints the Active Command List automatically following each run, listing the commands you issued and the operand values that were in effect during the run. An example of this report is shown in Figure 20 on page 81. The report is divided into two major sections: Selection Commands Issued and Process Commands Issued. In addition, these two categories are subdivided into three columns that provide detailed information about each command:

- Command Number
- Command Type
- Operands.

The Log Compare Utility assigns sequential numbers that appear in the **Command Number** column to each instance of the EXCLUDE, SELECT, START, SYNCPOINT, CHECKONLY, and MASK commands. Each command has its own sequence. Since you can enter each of these commands more than once, you can use these command numbers to determine exactly which command created the results you see on the report. The numbers for these commands also appear in the Complete Records Report, the Compare List Report, and the Summary Report.

The **Command Type** column lists the name of each command; the **Operands** column lists the operands and operand defaults associated with each command. When you code commands or operands with single character abbreviations, these columns show the exact commands and operands that were used by the Log Compare Utility.

For more information about coding control commands with single character abbreviations, refer to "Coding the control commands" on page 91.

Complete Records List

When you request the Complete Records List with the REPORT control command, the Log Compare Utility prints one report for each device being compared. An example of this report is shown in Figure 21 on page 81. The report lists every DSPY record in the log data set by log sequence number and indicates whether the record was used in the comparing process. If a record was not used in the process, an explanation is also included.

This report provides two major sections, MASTER Records and TEST Records, which are divided into four columns that provide detailed information about each record:

- Sequence Number
- MSGTXT
- Usage
- Reason.

During the simulation, WSim assigns sequential numbers, by device, to each DSPY record it creates. The **Sequence Number** column provides a list of each DSPY record and its assigned number. The utility uses the sequence numbers on each report, except the Active Command List, to identify a particular DSPY record.

The **MSGTXT** column lists the name of the message generation deck being processed when WSim logged the DSPY record. You can refer to this column when you debug a simulation and need to know where the DSPY records originated.

The **Usage** column lists whether the Log Compare Utility used the DSPY record during the comparing process. If the column specifies USED, the DSPY record was compared. If the column specifies NOT USED, the record was not compared.

In the **Reason** column, the Log Compare Utility provides an explanation about why certain DSPY records were or were not used in the compare process. This column might indicate that a SELECT command chose a record for comparison, for example, or that a specific record was selected during synchronization because it met the criteria established by a SYNCPOINT command.

When the Log Compare Utility does not use a record, the Usage column indicates that an EXCLUDE command excluded a record from the comparing process. In some cases, however, no explanation is provided. When the comparing process is aborted, for example, the Complete Records List places a message stating RUN ABORTED AT THIS TIME DUE TO NUMBER OF ERRORS after the last record used. This message supplies an explanation for all following records, showing NOT USED in the Usage column and no additional explanation in the Reason column.

To receive the Complete Records List following a run, specify the REPORT command with the RECORDS operand. Refer to “REPORT command” on page 105 for information about the REPORT command.

Compare List

When you request the Compare List, the Log Compare Utility prints a separate report for each device being compared. An example of this report is shown in Figure 22 on page 82. Each report matches the sequence numbers of DSPY records from the MASTER log data set against DSPY records from the TEST log data set and describes the results of the comparing process.

Each report contains the following columns of information:

- MASTER Sequence Number
- TEST Sequence Number
- Checkonly
- Mask
- All Mask

- Result
- Reason for Difference.

The **MASTER Sequence Number** and the **TEST Sequence Number** columns identify the sequence numbers of the MASTER and TEST DSPY records used for an individual comparison. However, the sequence number listed for the TEST DSPY record may not always be the same as the MASTER record's number. Following synchronization, for example, the Log Compare Utility may resume the comparing process using a MASTER DSPY Record with sequence number 20 and a TEST DSPY record with sequence number 44.

The **Checkonly** column indicates which CHECKONLY command, if any, was used for this individual comparison. The number that appears in this column is identical to the number listed in the Command Number column of the Active Command List. For more information about the Active Command List, refer to "Active Command List" on page 73.

The **Mask** column indicates which MASK command, if any, was used for this individual comparison. Like the number listed in the Checkonly column, the number appearing in the Mask column is identical to the number listed in the Command Number column on the Active Command List.

If you code a Mask command with only the MASKLOC operand, the utility masks every panel used at the specified location. This is called an ***all mask*** operation. When you specify "all mask," an asterisk (*) appears in the **All Mask** column. For more information about the MASKLOC operand, refer to "MASK command" on page 103.

When the Log Compare Utility compares two DSPY records, one of two possible results are listed in the **Result** column: EQUAL or NOT EQUAL. EQUAL indicates that the utility found the two DSPY records to be identical relative to how you coded the commands. NOT EQUAL, however, means that some difference was detected between the records. Whenever NOT EQUAL appears, the utility provides an explanation of the difference between the two records in the Reason for Difference column.

The **Reason for Difference** column describes a comparison that resulted in a NOT EQUAL message in the Result column. Although the most commonly occurring reason for a difference can be DATA DIFFERENCE DETECTED, other reasons may include messages for unequal panel size or an attribute difference. When the Compare List shows that a data or attribute difference existed between two records, you can obtain additional information from the Differences Report, as discussed in the following section.

To receive the Compare List following a run, specify the REPORT command with the COMPARES operand. Refer to "REPORT command" on page 105 for information about the REPORT command.

Differences Report

The Log Compare Utility prints a Differences Report for the first occurrence of a difference between DSPY records. An example of this report is shown in Figure 24 on page 84. If the records were unequal because of a difference in data or cursor position, the utility prints a screen image for each of the MASTER and TEST records. If the records were unequal because of a difference in attributes, the utility prints the attribute values and a description of the difference in a table. If no difference is found, the Log Compare Utility prints a Differences Report indicating no differences.

The header provided on the Differences Report lists the sequence number for both the MASTER and TEST DSPY records and provides the row and column panel location of the difference.

You can obtain a Differences Report in either of the following two ways:
- Do not specify the REPORT command
- Specify the REPORT command with the DIFFERENCES operand.

Refer to “REPORT command” on page 105 for information about the REPORT command.

Summary Report

The Summary Report lists the overall results of the run by device, including the following information:
- The number of MASTER and TEST DSPY records processed
- The number of records compared
- The number of differences detected during the run
- Whether the utility attempted to synchronize records
- Whether the utility aborted the run before completion
- Whether synchronization was attempted.

If the Log Compare Utility attempted to synchronize the two log data sets, the Summary Report also lists the SYNCPOINT command used and the sequence number for the MASTER and TEST DSPY records. An example of this report is shown in Figure 27 on page 89.

You can obtain a Summary Report in either of the following two ways:
- Do not specify the REPORT command
- Specify the REPORT command with the SUMMARY operand.

Refer to “REPORT command” on page 105 for information about the REPORT command.

Running the Log Compare Utility

The Log Compare Utility requires information about how you want the MASTER and TEST DSPY records compared. You supply this information to the utility with control commands, which you can provide in an input file or enter from the console, and with the JCL or TSO CLIST statements you use to run the utility. These statements provide the utility with the following locations:
- MASTER log data set
- TEST log data set
- Command input file
- Log Compare Utility program
- The printer

The following sections describe more information about the Log Compare Utility:
- Running the Log Compare Utility using the WSim/ISPF Interface
- Execution parameters

- Examples of JCL and a TSO CLIST
- Output generated by a sample control command file.

Using the WSim/ISPF Interface

You can run the Log Compare Utility from the WSim/ISPF Interface. To do this, follow these steps:

1. Invoke the WSim/ISPF Interface main panel from ISPF. The method you use to do this depends on how the WSim/ISPF Interface application is installed at your site. If you are not sure how to do this, see your system programmer for assistance.
2. Select option 9 from the WSim/ISPF Interface main panel and press **Enter**. The Compare Logged Display Data panel is displayed.

   Note: You can also type "COMPARE" on the command line and press **Enter** to display this panel.
3. Fill in the appropriate information for this panel and press **Enter** to run the Log Compare Utility.

For more information on the WSim/ISPF Interface, see Chapter 2, "Running WSim with the WSim/ISPF Interface," on page 5.

Using Log Compare Utility execution parameters

You can enter the following execution parameters, which are optional, in the PARM field for the JCL EXEC statement or the TSO CLIST statement when you run the Log Compare Utility.

CONSOLE
Specifies that the utility issue a WTOR for the input control commands. If you do not specify CONSOLE, the utility reads the control commands from the SYSIN data set.

PRMODE=(<u>LINE</u>|SOSI1|SOSI2)
Indicates the type of system printer support available. When printing DBCS data in 3270 log display records, specifying PRMODE indicates whether the system printer leaves a space when an SO or SI character is encountered. DBCS data in console records is printed as-is.

You can specify the following values for PRMODE:

LINE	Indicates that the system printer cannot print DBCS data. This is the default value.
SOSI1	Indicates that the system printer leaves a space when an SO or SI character is encountered. When you specify PRMODE=SOSI1, extra blank characters are not inserted into print records containing DBCS data delimited by SO and SI characters.
SOSI2	Indicates that the system printer does not leave a space when an SO or SI character is printed. When you specify PRMODE=SOSI2, extra blank characters are inserted into print records containing DBCS data delimited by SO and SI characters. These blanks maintain the character spacing of simulated 3270 devices that display SO and SI characters as blanks.

You must specify PRMODE=SOSI1 or PRMODE=SOSI2 to print DBCS data. If you specify PRMODE=LINE, or do not specify PRMODE, DBCS data in 3270 log display records is formatted as SBCS data. This allows screen images containing DBCS data to be printed on a non-DBCS printer.

PRTLNCNT=*integer*
Specifies the maximum number of lines that the utility prints on a page of output before ejecting to a new page. *integer* can be a value from 35 to 255; the default value is 60.

ROUTCDE=(*n,n...***)**
Specifies the message routing codes that the utility uses in writing Log Compare Utility messages to the operator. Each *n* is a system routing code that defines a console destination for every WTO and WTOR written by the Log Compare Utility. The value for *n* is an integer from 1 to 16; the default value is 8.

Data set requirements

The following data sets are required to run the Log Compare Utility.

Data Set	Function
STEPLIB DD	Defines the data set containing the WSim host processor modules.
SYSPRINT DD	Defines the output printer. SYSPRINT records may be either fixed or variable length. For fixed length records, logical record lengths of 133 to 256 are accepted. For variable length records, logical record lengths of 137 to 260 are accepted. In either case, the maximum length of noncontrol printed data is 255 bytes. The default is fixed 133 byte length with blocking supported. Record lengths larger than 133 are utilized when PRMODE=SOSI1 or PRMODE=SOSI2 is specified. A larger record length allows additional DBCS data to be printed when the formatted 3270 log display images are printed.
SYSIN DD	Defines the control command input file.
MASTLDS DD	Specifies the name of the MASTER log data set.
TESTLDS DD	Specifies the name of the TEST log data set.

Using JCL

The example below shows the JCL you can use to run the Log Compare Utility on MVS.

```
//*************************************************************************
//*                       Log Compare Utility JCL                        *
//*                                                                      *
//*  The following example illustrates the JCL you can use to run the    *
//*  Compare Utility in an MVS environment.                              *
//*                                                                      *
//*************************************************************************
//COMPJOB  JOB                                Names the JCL job stream.
//ITPCOMP  EXEC PGM=ITPCOMP,REGION=4096K,  Specifies the program name.
//         PARM='PRTLNCNT=60'
//STEPLIB  DD DSN=WSIM.SITPLOAD,DISP=SHR   Defines the data set that
//*                                        contains the WSim host
//*                                        processor modules.
//SYSPRINT DD SYSOUT=A                     Defines the printer output.
//MASTLDS  DD DSN=COMPARE.MASTER,DISP=SHR  Defines the MASTER log data set.
//TESTLDS  DD DSN=COMPARE.TEST,DISP=SHR    Defines the TEST log data set.
//SYSIN DD *
  RUN
  END
/*
```

Using a TSO CLIST

The following shows an example of a TSO CLIST to run the Log Compare Utility under TSO.

```
/****************************************************************
/*            Log Compare Utility TSO CLIST                    *
/* The following example illustrates a TSO CLIST you can use    *
/* the Log Compare Utility.                                     *
/****************************************************************
FREE    DDNAME(SYSPRINT SYSIN MASTLDS TESTLDS)
ALLOC   DDNAME(SYSPRINT) SYSOUT(A)
ALLOC   DDNAME(SYSIN)  DATASET('COMPARE.SYSIN') SHR
ALLOC   DDNAME(MASTLDS)  DATASET('COMPARE.MASTER') SHR
ALLOC   DDNAME(TESTLDS)  DATASET('COMPARE.TEST') SHR
CALL    'WSIM.SITPLOAD(ITPCOMP)' 'PRTLNCNT=60'
FREE    DDNAME(SYSPRINT SYSIN MASTLDS TESTLDS)
```

Understanding sample output

The following pages contain some examples of the reports created when you use the WSim/ISPF Interface, JCL or a TSO CLIST to run the Log Compare Utility with the input command file shown in Figure 19. Figure 20 on page 81 through Figure 27 on page 89 show examples of the output.

```
NTWRK    SAMPNET
VTAMAPPL WSIM1,WSIM2
LU       VAPPL13,VAPPL23
START    DATA=(Logon),
         LOC=(2,1),
         SCAN=80
MASK     MASKLOC=(4,41,4)
EXCLUDE  DATA=(Set A),
         LOC=(1,76)
CHECKONLY DATA=(Logon),
          SCAN=YES,
          CHECKLOC=(1,76,5)
ATTRIBUTE
CHARATTR
CURSOR
REPORT R,S,C,D
RUN
```

Figure 19. Control commands to specify Log Compare Utility output

The Log Compare Utility uses two types of control commands: process and selection. Refer to Chapter 7, "Specifying Log Compare Utility control commands," on page 91 for information about the two types of commands.

The following five commands are the selection commands used in Figure 19:

NTWRK SAMPNET
: Limits the compare process to devices under the network SAMPNET only.

VTAMAPPL WSIM1,WSIM2
: Limits the compare process to devices under the VTAM application WSIM1 in the MASTER log data set and VTAM application WSIM2 in the TEST log data set.

LU VAPPL13,VAPPL23
: Limits the compare process to the logical unit VAPPL13 in the MASTER log data set and logical unit VAPPL23 in the TEST log data set.

START DATA=(Logon),LOC=(2,1),SCAN=80
: Describes the screen on which the utility should begin the comparison process. In this example, the utility finds the first DSPY record in the MASTER and TEST log data sets that contains the word "Logon", beginning the search on row 2 column 1 and searching the next 80 characters.

EXCLUDE DATA=(Set A),LOC=(1,76)
: In this example there are two different sets of screens: Set A and Set B. All screens are marked either Set A or Set B in row 1, column 76. In this case, all of the DSPY records for Set A will be excluded from the comparison.

The following six commands are the selection commands used in Figure 19 on page 79:

MASK MASKLOC=(4,41,4)
: This is what is known as an ALL MASK because the DATA option is not coded. As a result, this MASK applies to ALL DSPY records. In this case, the utility ignores 4 characters of data starting in row 4 column 41. The MASK statement is often used to mask time stamps.

CHECKONLY DATA=(Logon),SCAN=YES,CHECKLOC=(1,76,5)
: This command tells the utility that for certain screens, only one field is to be compared, not the whole screen. In this case, on all screens that contain the word "Logon", the utility compares only the field located in row 1 column 76 for 5 characters.

  Note: You can code several CHECKLOC options for one CHECKONLY statement.

ATTRIBUTE
: This specifies that attributes are to be compared.

CHARATTR
: This specifies that character attributes are to be compared.

CURSOR
: This specifies that the cursor position is to be compared.

REPORT
: This specifies that all four reports are to be run (Records, Compares, Differences, Summary).

These reports are discussed in the following sections.

Active Command List

Figure 20 on page 81 shows an example of the Active Commands List, which prints automatically when you run the Log Compare Utility. This list shows which commands were active for the results printed in the other Log Compare Utility reports. The list is divided into two major groups: selection commands and process commands. Refer to Chapter 7, "Specifying Log Compare Utility control commands," on page 91 for information about the two types of commands.

```
------------------------------------------------------------------------------------------------------------------------------------
                                                  Active Command List
------------------------------------------------------------------------------------------------------------------------------------
   Selection Commands Issued
   -------------------------
        Command    Command
         Number     Type        Operands
        -------  -----------   ------------------------------------------------------------
                   NTWRK        SAMPNET
                   VTAMAPPL     WSIM1,WSIM2
                   LU           VAPPL13,VAPPL23
              1    START        DATA=(Logon),
                                LOC=(2,1),
                                SCAN=80
              1    EXCLUDE      DATA=(Set A),
                                LOC=(1,76),
                                SCAN=1
   Process Commands Issued
   -----------------------
        Command    Command
         Number     Type        Operands
        -------  -----------   ------------------------------------------------------------
                   ATTRIBUTE
                   CHARATTR
              1    CHECKONLY    DATA=(Logon),
                                LOC=(1,1),
                                SCAN=32767,
                                CHECKLOC=(1,76,5)
                   CURSOR
            ALL    MASK         MASKLOC=(4,41,4)
                   REPORT       RECORDS,COMPARES,DIFFERENCES,SUMMARY
```

Figure 20. Example of an active command list

Complete Records List

The Log Compare Utility Complete Records List has two lists: one for the MASTER and one for the TEST log data set. Figure 21 shows the panels, identified by their sequence numbers, that the Log Compare Utility compared for an example simulation.

```
------------------------------------------------------------------------------------------------------------------------------------
                                                  Complete Records List
Master:  NETWORK     SAMPNET                                                          Test:  NETWORK     SAMPNET
         VTAMAPPL    WSIM1                                                                   VTAMAPPL    WSIM2
         DEV/LU      VAPPL13-00001                                                           DEV/LU      VAPPL23-00001
------------------------------------------------------------------------------------------------------------------------------------
  MASTER Records
  --------------
    Sequence
     Number      MSGTXT        Usage         Reason
    --------   ----------    ----------    -------------------------------------------------------------------------------
          0     SLU3          Used          Met START command; met CHECKONLY #1
          1     SLU3          Not Used      Met EXCLUDE #1
          2     SLU3          Used          Met ALL MASK
          3     SLU3          Not Used      Met EXCLUDE #1
          4     SLU3          Not Used      Met EXCLUDE #1
          5     SLU3          Used          Met ALL MASK
          6     SLU3          Used          Met ALL MASK
          7     SLU3          Used          Met ALL MASK
          8     SLU3          Not Used      Met EXCLUDE #1
          9     SLU3          Not Used      Met EXCLUDE #1
         10     SLU3          Used          Met ALL MASK
  TEST Records
  ------------
    Sequence
     Number      MSGTXT        Usage         Reason
    --------   ----------    ----------    -------------------------------------------------------------------------------
          0     SLU           Used          Met START command; met CHECKONLY #1
          1     SLU           Not Used      Met EXCLUDE #1
          2     SLU           Used          Met ALL MASK
          3     SLU           Not Used      Met EXCLUDE #1
          4     SLU           Not Used      Met EXCLUDE #1
          5     SLU           Used          Met ALL MASK
          6     SLU           Used          Met ALL MASK
          7     SLU           Used          Met ALL MASK
          8     SLU           Not Used      Met EXCLUDE #1
          9     SLU           Not Used      Met EXCLUDE #1
         10     SLU           Used          Met ALL MASK
```

Figure 21. Example of a complete records list (MASTER and TEST)

Compare List

Figure 22 on page 82 is an example of the Log Compare Utility Compare List. It shows the comparison of DSPY records from only those panels used, including sequence numbers 0, 2, 5, 6, 7, and 10. The Checkonly column shows that CHECKONLY statement number one was used on panel zero. The ALL Mask column shows that an ALL Mask was applied to the rest of the panels. (Refer to Figure 20 to see a listing of the process commands issued.) The Result column shows the comparison of five panels were not equal and shows the reason for those differences.

```
------------------------------------------------------------------------------------------------------------------------------------
                                                  Compare List
Master:  NETWORK     SAMPNET                                                               Test:  NETWORK     SAMPNET
         VTAMAPPL    WSIM1                                                                        VTAMAPPL    WSIM2
         DEV/LU      VAPPL13-00001                                                                DEV/LU      VAPPL23-00001
------------------------------------------------------------------------------------------------------------------------------------
    MASTER            TEST                              ALL
Sequence Number   Sequence Number   Checkonly  Mask   Mask    Result     REASON FOR DIFFERENCE
---------------   ---------------   ---------  -----  ----  ---------    -----------------------------------------------------------
         0                 0              1                   Equal
         2                 2                            *    Not Equal   Character Attribute Difference Detected
         5                 5                            *    Not Equal   Attribute Difference Detected
         6                 6                            *    Not Equal   Data Difference Detected
         7                 7                            *    Not Equal   Cursor Position Difference Detected
        10                10                            *    Not Equal   Data Difference Detected
```

Figure 22. Example of a compare list

Differences Report

The Log Compare Utility Differences Report describes in detail the differences detected between the DSPY records found in the MASTER and TEST log data sets. Figure 23 shows an example of a Differences report produced when a base or character attribute difference exists between the MASTER DSPY record and the TEST DSPY record. In this particular example, the character attribute difference was between REVERSE VIDEO in the MASTER DSPY record and UNDERLINED in the TEST DSPY record. Also, a base attribute difference exists. The MASTER DSPY record contains a value of X'F8' and the TEST DSPY record contains a value of X'F4'. In other words, the MASTER DSPY record has high intensity defined and the TEST DSPY record does not have high intensity defined. Only the first difference identified on a panel is listed in the report. The locations of other differences, if any, are not noted by row and column in the report.

```
------------------------------------------------------------------------------------------------------------------------------------
                                                  Differences Report
Master:  NETWORK     SAMPNET                                                               Test:  NETWORK     SAMPNET
         VTAMAPPL    WSIM1                                                                        VTAMAPPL    WSIM2
         DEV/LU      VAPPL13-00001                                                                DEV/LU      VAPPL23-00001
------------------------------------------------------------------------------------------------------------------------------------
Character Attribute Error  MASTER Sequence Number 2        TEST Sequence Number 2
            Highlight        Color      Character Set
          -------------    ---------    -------------
MASTER    REVERSE VIDEO    YELLOW            BASE
TEST      UNDERLINED       YELLOW            BASE

------------------------------------------------------------------------------------------------------------------------------------
                                                  Differences Report
Master:  NETWORK     SAMPNET                                                               Test:  NETWORK     SAMPNET
         VTAMAPPL    WSIM1                                                                        VTAMAPPL    WSIM2
         DEV/LU      VAPPL13-00001                                                                DEV/LU      VAPPL23-00001
------------------------------------------------------------------------------------------------------------------------------------
Base Attribute Error           MASTER Sequence Number 5        TEST Sequence Number 5
          Value    Un/Protected    Alpha/Numeric    High Intensity    Selector Pen Detectable    Non-Display    Modified Data Tag
          -----    ------------    -------------    --------------    -----------------------    -----------    -----------------
MASTER    'F8'X       PROTECTED     AUTO SKIP            YES                    YES                   NO               OFF
TEST      'F4'X       PROTECTED     AUTO SKIP             NO                    YES                   NO               OFF
```

Figure 23. Example of a differences report for character and base attribute differences

The Differences Reports for data and cursor position differences have two parts: the formatted screen image of the MASTER DSPY record and the formatted screen image of the TEST DSPY record. Figure 24 on page 84 shows the Differences Report for the MASTER DSPY and TEST DSPY records detecting upper and lowercase differences. The location of the first difference detected is printed in the upper right corner of both parts of the report. In the following example, the difference is located at (15,46), where the MASTER DSPY record contains an "a" (lowercase) and the TEST DSPY record contains an "A" (uppercase). Only the first difference identified on a panel is listed in the report. The locations of other differences, if any, are not noted by row and column in the report.

The example shows the report with data differences on only one panel. If 50 panels were different, this report would be considerably longer. Figure 25 on page 86 shows the Differences Report for the MASTER DSPY and TEST DSPY records detecting cursor position differences. In the following example, the cursor is located at (15,42) in the MASTER DSPY record and at (18,42) in the TEST DSPY record.

```
------------------------------------------------------------------------------------------------------------------------------------
                                                  Differences Report
Master:  NETWORK     SAMPNET                                                              Test:  NETWORK     SAMPNET
         VTAMAPPL    WSIM1                                                                       VTAMAPPL    WSIM2
         DEV/LU      VAPPL13-00001                                                               DEV/LU      VAPPL23-00001
------------------------------------------------------------------------------------------------------------------------------------
MASTER Screen Image        MASTER Sequence Number 6         TEST Sequence Number 6        Location of Difference    (15,46)

             1         2         3         4         5         6         7         8
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
    --------------------------------------------------------------------------------
  1|                                                                           Set B|  1
  2|                              Log Compare Screen                                |  2
  3|                                                                                |  3
  4|                                   Mask mmmm                                    |  4
  5|                                                                                |  5
  6|                          Lower / Uppercase Example                             |  6
  7|                                                                                |  7
  8|                                                                                |  8
  9|                                                                                |  9
 10|                                                                                | 10
 11|                                                                                | 11
 12|                                                                                | 12
 13|                                                                                | 13
 14|                                                                                | 14
 15|               This is data field one: aaaaaaa                                  | 15
 16|                                                                                | 16
 17|                                                                                | 17
 18|               This is data field two:  aaaaa                                   | 18
 19|                                                                                | 19
 20|                                                                                | 20
 21|                                                                                | 21
 22|                                                                                | 22
 23|                                                                                | 23
 24|                                                                                | 24
    --------------------------------------------------------------------------------
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
             1         2         3         4         5         6         7         8
      CURSOR: ROW( 18) COLUMN( 46)   AID: ENTER KEY                  WHEN LOGGED: END OF MSG GEN
      DIMENSIONS: ( 24, 80)
------------------------------------------------------------------------------------------------------------------------------------
                                                  Differences Report        (continued)
Master:  NETWORK     SAMPNET                                                              Test:  NETWORK     SAMPNET
         VTAMAPPL    WSIM1                                                                       VTAMAPPL    WSIM2
         DEV/LU      VAPPL13-00001                                                               DEV/LU      VAPPL23-00001
------------------------------------------------------------------------------------------------------------------------------------
TEST Screen Image          MASTER Sequence Number 6         TEST Sequence Number 6        Location of Difference    (15,46)

             1         2         3         4         5         6         7         8
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
    --------------------------------------------------------------------------------
  1|                                                                           Set B|  1
  2|                              Log Compare Screen                                |  2
  3|                                                                                |  3
  4|                                   Mask 6666                                    |  4
  5|                                                                                |  5
  6|                          Lower / Uppercase Example                             |  6
  7|                                                                                |  7
  8|                                                                                |  8
  9|                                                                                |  9
 10|                                                                                | 10
 11|                                                                                | 11
 12|                                                                                | 12
 13|                                                                                | 13
 14|                                                                                | 14
 15|               This is data field one: aaaaaaA                                  | 15
 16|                                                                                | 16
 17|                                                                                | 17
 18|               This is data field two:  AAAAA                                   | 18
 19|                                                                                | 19
 20|                                                                                | 20
 21|                                                                                | 21
 22|                                                                                | 22
 23|                                                                                | 23
 24|                                                                                | 24
    --------------------------------------------------------------------------------
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
             1         2         3         4         5         6         7         8
      CURSOR: ROW( 18) COLUMN( 46)   AID: ENTER KEY                  WHEN LOGGED: END OF MSG GEN
      DIMENSIONS: ( 24, 80)
```

Figure 24. Example of a differences report with a upper/lowercase difference

```
------------------------------------------------------------------------------------------------------------------------------------
                                                  Differences Report
Master:  NETWORK     SAMPNET                                                                Test:  NETWORK     SAMPNET
         VTAMAPPL    WSIM1                                                                         VTAMAPPL    WSIM2
         DEV/LU      VAPPL13-00001                                                                 DEV/LU      VAPPL23-00001
------------------------------------------------------------------------------------------------------------------------------------
MASTER Screen Image        MASTER Sequence Number 7         TEST Sequence Number 7        Cursor Position Difference

             1         2         3         4         5         6         7         8
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
    --------------------------------------------------------------------------------
  1|                                                                           Set B|  1
  2|                              Log Compare Screen                                |  2
  3|                                                                                |  3
  4|                                   Mask kkkk                                    |  4
  5|                                                                                |  5
  6|                            Cursor Position Example                             |  6
  7|                                                                                |  7
  8|                                                                                |  8
  9|                                                                                |  9
 10|                                                                                | 10
 11|                                                                                | 11
 12|                                                                                | 12
 13|                                                                                | 13
 14|                                                                                | 14
 15|               This is position one ==>  aaaaaa                                 | 15
 16|                                                                                | 16
 17|                                                                                | 17
 18|               This is position two ==>  aaaaaa                                 | 18
 19|                                                                                | 19
 20|                                                                                | 20
 21|                                                                                | 21
 22|                                                                                | 22
 23|                                                                                | 23
 24|                                                                                | 24
    --------------------------------------------------------------------------------
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
             1         2         3         4         5         6         7         8
      CURSOR: ROW( 15) COLUMN( 42)   AID: ENTER KEY                  WHEN LOGGED: END OF MSG GEN
      DIMENSIONS: ( 24, 80)
------------------------------------------------------------------------------------------------------------------------------------
                                                  Differences Report        (continued)
Master:  NETWORK     SAMPNET                                                                Test:  NETWORK     SAMPNET
         VTAMAPPL    WSIM1                                                                         VTAMAPPL    WSIM2
         DEV/LU      VAPPL13-00001                                                                 DEV/LU      VAPPL23-00001
------------------------------------------------------------------------------------------------------------------------------------
TEST Screen Image          MASTER Sequence Number 7         TEST Sequence Number 7        Cursor Position Difference

             1         2         3         4         5         6         7         8
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
    --------------------------------------------------------------------------------
  1|                                                                           Set B|  1
  2|                              Log Compare Screen                                |  2
  3|                                                                                |  3
  4|                                   Mask 7777                                    |  4
  5|                                                                                |  5
  6|                            Cursor Position Example                             |  6
  7|                                                                                |  7
  8|                                                                                |  8
  9|                                                                                |  9
 10|                                                                                | 10
 11|                                                                                | 11
 12|                                                                                | 12
 13|                                                                                | 13
 14|                                                                                | 14
 15|               This is position one ==>  aaaaaa                                 | 15
 16|                                                                                | 16
 17|                                                                                | 17
 18|               This is position two ==>  aaaaaa                                 | 18
 19|                                                                                | 19
 20|                                                                                | 20
 21|                                                                                | 21
 22|                                                                                | 22
 23|                                                                                | 23
 24|                                                                                | 24
    --------------------------------------------------------------------------------
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
             1         2         3         4         5         6         7         8
      CURSOR: ROW( 18) COLUMN( 42)   AID: ENTER KEY                  WHEN LOGGED: END OF MSG GEN
      DIMENSIONS: ( 24, 80)
```

Figure 25. Example of a differences report with a cursor position difference

Figure 26 on page 88 shows the Differences Report for the MASTER DSPY and TEST DSPY records detecting data differences. In the following example, the difference is located at (18,46). The MASTER DSPY record contains “aaaaaa” and the TEST DSPY record contains “zzzzzz”.

```
------------------------------------------------------------------------------------------------------------------------------------
                                                  Differences Report
Master:  NETWORK     SAMPNET                                                               Test:  NETWORK     SAMPNET
         VTAMAPPL    WSIM1                                                                        VTAMAPPL    WSIM2
         DEV/LU      VAPPL13-00001                                                                DEV/LU      VAPPL23-00001
------------------------------------------------------------------------------------------------------------------------------------
MASTER Screen Image        MASTER Sequence Number 10        TEST Sequence Number 10       Location of Difference    (18,36)

             1         2         3         4         5         6         7         8
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
    --------------------------------------------------------------------------------
  1|                                                                           Set B|  1
  2|                              Log Compare Screen                                |  2
  3|                                                                                |  3
  4|                                   Mask qqqq                                    |  4
  5|                                                                                |  5
  6|                                 Data Example                                   |  6
  7|                                                                                |  7
  8|                                                                                |  8
  9|                                                                                |  9
 10|                                                                                | 10
 11|                                                                                | 11
 12|                                                                                | 12
 13|                                                                                | 13
 14|                                                                                | 14
 15|               This is field one:  aaaaaa                                       | 15
 16|                                                                                | 16
 17|                                                                                | 17
 18|               This is field two:  aaaaaa                                       | 18
 19|                                                                                | 19
 20|                                                                                | 20
 21|                                                                                | 21
 22|                                                                                | 22
 23|                                                                                | 23
 24|                                                                                | 24
    --------------------------------------------------------------------------------
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
             1         2         3         4         5         6         7         8
      CURSOR: ROW( 15) COLUMN( 36)   AID: ENTER KEY                  WHEN LOGGED: END OF MSG GEN
      DIMENSIONS: ( 24, 80)
------------------------------------------------------------------------------------------------------------------------------------
                                                  Differences Report        (continued)
Master:  NETWORK     SAMPNET                                                               Test:  NETWORK     SAMPNET
         VTAMAPPL    WSIM1                                                                        VTAMAPPL    WSIM2
         DEV/LU      VAPPL13-00001                                                                DEV/LU      VAPPL23-00001
------------------------------------------------------------------------------------------------------------------------------------
TEST Screen Image          MASTER Sequence Number 10        TEST Sequence Number 10       Location of Difference    (18,36)

             1         2         3         4         5         6         7         8
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
    --------------------------------------------------------------------------------
  1|                                                                           Set B|  1
  2|                              Log Compare Screen                                |  2
  3|                                                                                |  3
  4|                                   Mask 0000                                    |  4
  5|                                                                                |  5
  6|                                 Data Example                                   |  6
  7|                                                                                |  7
  8|                                                                                |  8
  9|                                                                                |  9
 10|                                                                                | 10
 11|                                                                                | 11
 12|                                                                                | 12
 13|                                                                                | 13
 14|                                                                                | 14
 15|               This is field one:  aaaaaa                                       | 15
 16|                                                                                | 16
 17|                                                                                | 17
 18|               This is field two:  zzzzzz                                       | 18
 19|                                                                                | 19
 20|                                                                                | 20
 21|                                                                                | 21
 22|                                                                                | 22
 23|                                                                                | 23
 24|                                                                                | 24
    --------------------------------------------------------------------------------
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
             1         2         3         4         5         6         7         8
      CURSOR: ROW( 15) COLUMN( 36)   AID: ENTER KEY                  WHEN LOGGED: END OF MSG GEN
      DIMENSIONS: ( 24, 80)
```

Figure 26. Example of a differences report with a data difference

Summary Report

Figure 27 shows an example of the Log Compare Utility Summary Report.

```
------------------------------------------------------------------------------------------------------------------------------------
                                                  Summary Report
------------------------------------------------------------------------------------------------------------------------------------
                                                                                                   Synchronization        Sequence
                                          Result #Records Processed #Records #Differences   Run             SYNCPOINT     Number
Resource                                   (RC)   MASTER     TEST   Compared   Detected   Aborted Attempted Command Used MASTER TEST
----------------------------------------- ------ --------  -------- -------- ------------ ------- --------- ------------ -----------
NETWORK     SAMPNET
VTAMAPPL    WSIM1,WSIM2
DEV/LU      VAPPL13-00001,VAPPL23-00001       4       11       11        6         5      NO       NO
------------------------------------------------------------------------------------------------------------------------------------
```

Figure 27. Example of a summary report

Understanding Log Compare Utility return codes

After running, the Log Compare Utility sets a return code to indicate the status of the execution. The Log Compare Utility can return the following codes:

Code	**Meaning**
0	The run completed with no errors.
4	The run completed with no errors. A difference was found for at least one resource.
8	The run completed with no errors. A difference was found for at least one resource and synchronization was attempted.
12	The run completed with no errors. A difference was found for at least one resource and the run was aborted.
16	An invalid command was specified.
20	Storage was not available for Log Compare Utility execution.
24	The SYSPRINT data set failed to open.
28	An invalid execution parameter was specified.
32	The MASTLDS data set failed to open.
36	The TESTLDS data set failed to open.
40	No problems with execution occurred, but no records were compared for at least one run. (A run contains all of the output generated from a single RUN command.)

Return codes 0, 4, 8, 12, and 40 are also used in the Summary Report for each device or LU. However, instead of the return code referring to the entire run, it refers to just the specific device or LU.

The overall return code is 40 only when all devices and LUs in a run have codes of 40.

Chapter 7. Specifying Log Compare Utility control commands

The following sections provide a list of the control commands and operands you can use to operate the Log Compare Utility. To help you use this information, the next sections explain the requirements to enter the coding, the coding conventions for control commands, and the two categories of control commands that you can use to specify the DSPY records or portions of records to be compared during the comparing process:

- Selection commands
- Process commands.

Coding the control commands

When you code Log Compare Utility control commands and operands, you can enter a substring of the full keyword as an abbreviation. For example,

```
S D=(a),L=(1,1),S=Y
```

is equivalent to

```
SELECT DATA=(a),LOC=(1,1),SCAN=YES
```

However, the character string you use must be long enough to identify the command or operand uniquely. Although the character string "S" could represent either SELECT or START, the Log Compare Utility identifies the command alphabetically as SELECT. To abbreviate START you must code "ST" instead of simply "S".

You must code all control commands in uppercase when entered using the control command input file.

Note: The other WSim utilities do not recognize abbreviated commands, except for the Loglist Utility and Response Time Utility, which recognize a few. Refer to Chapter 5, "Specifying loglist control commands," on page 45 for a listing of the Loglist Utility control commands.

You can also include comments on all control commands that contain at least one operand. Leave one blank space after the operand, and then enter the comment. The Log Compare Utility ignores the comments but lists them with the control commands as you entered them.

Understanding control command coding conventions

The control commands for the Log Compare Utility use the same coding conventions as the Loglist Utility commands. Refer to "Understanding control command coding conventions" on page 45 for information about the coding conventions.

Selection commands

Selection commands define which records the Log Compare Utility compares during a run. During the comparing process, the utility does not consider a DSPY record for comparison unless the record meets the qualifications set by the selection commands. For example, when you specify the DEV selection command, the Log Compare Utility compares only DSPY records for the named device.

DEV command

```
DEV devid1[,devid2]
```

The DEV command specifies the name of the device that the Log Compare Utility uses for the comparing process. You may enter this command multiple times for each run to process multiple devices. If you omit this command entirely, all devices found by the utility in the log data set are considered valid for the comparing process.

If the DEV command does not follow a valid TCPIP or VTAMAPPL command, the Log Compare Utility uses the named device on all TCP/IP connections and VTAMAPPLs in the comparing process. If the DEV command does follow a valid TCPIP or VTAMAPPL command, the Log Compare Utility uses the named device on only that TCP/IP connection or VTAMAPPL in the comparing process.

If you do not enter a TERM, DEV, or LU command following a TCPIP or VTAMAPPL command, the Log Compare Utility processes all devices or logical units for the specified TCP/IP connection or VTAMAPPL. If no resource commands are entered, all devices found in the log data set are valid for the comparing process.

devid1

Function: The *devid1* operand specifies the name of the device or logical unit (LU) used in the MASTER log data set.

Format: The value of the *devid1* operand is a 1- to 8-character name that matches the name coded on a WSim network definition DEV or LU statement.

devid2

Function: The *devid2* operand specifies the name of the device or LU used in the TEST log data set.

Format: The value of the *devid2* operand is a 1- to 8-character name that matches the name coded on a WSim network definition DEV or LU statement.

Default: If *devid2* is not specified, the name specified for *devid1* is used as the name of the device or LU used in the TEST log data set.

ERRCOUNT command

```
ERRCOUNT [ABORT=integer]
         [SYNCPOINT=integer]
```

The ERRCOUNT command specifies the number of differences that may occur before the utility performs the actions specified by the ABORT and SYNCPOINT operands. If you do not code either operand, the utility sets ABORT and SYNCPOINT to their default values.

ABORT=*integer*

Function: The ABORT operand specifies the total number of differences per device or LU that may occur before the utility aborts the comparing process for that device or LU.

Format: *integer* can be an integer from 1 to 32767.

Default: 10.

SYNCPOINT=*integer*

Function: The SYNCPOINT operand specifies the number of consecutive mismatches the utility may allow before trying to synchronize the log data sets.

Format: *integer* can be an integer from 1 to 255.

Default: 2.

EXCLUDE command

```
EXCLUDE DATA=(data)
       [,LOC=(row,col)]
       [,SCAN={YES|integer|1}]
```

The EXCLUDE command identifies specific records that the utility excludes from the comparing process. This command is mutually exclusive with the SELECT command. The EXCLUDE command is processed after the START command. Thus, the starting display record may be a record that is excluded.

To use this command, you specify a string of data that identifies the 3270 DSPY record or records to be excluded. You can specify the exact position of the data or have WSim scan the entire record.

DATA=(*data***)**

Function: The DATA operand specifies the data that identifies the record or records to be excluded. The Log Compare Utility requires this operand whenever you specify the EXCLUDE command.

Format: You can represent the data as EBCDIC, hexadecimal (enclosed in single quote marks), or a mixture of both; the data can be up to 20 bytes in length.

Note: SO and SI characters are removed from DBCS data entered and are not counted as part of the previously mentioned 20 bytes. To include an SO or SI character in the data for the data compare, add the hexadecimal equivalent of an SO (X'0E') or SI (X'0F') to the data.

LOC=(*row***,***col***)**

Function: The LOC operand indicates the exact panel position where the utility can find the data specified by the DATA operand.

Format: The values for *row* and *col* can be integers from 1 to 255.

Default: The default is 1 for both *row* and *col*.

SCAN={YES|*integer*|1}

Function: The SCAN operand specifies whether or not the utility is to scan the record sequentially for the data specified in the DATA operand. When you specify scanning, the utility searches the record starting at the location specified in the LOC operand. If you omit the LOC operand, the starting location is (1,1) of the screen image.

If you code YES for the SCAN operand, the utility searches the record from the position specified by the LOC operand to the end. If you specify an integer, the utility searches the number of positions specified by *integer*.

Format: *integer* can be an integer from 1 to 32767.

Default: 1.

LU command

```
LU lu1[-num1][,lu2[-num2]]
```

The LU operand specifies the name of the logical unit that the utility uses for the comparing process. You may enter this command multiple times for each run to process multiple LUs. If you omit this command, the utility considers all LUs found in the log data set valid for the comparing process.

If the LU command does not follow a valid TCPIP or VTAMAPPL command, the Log Compare Utility uses the named logical unit from TCP/IP connections and VTAMAPPLs in the comparing process. If the LU command does follow a valid TCPIP or VTAMAPPL command, the Log Compare Utility uses the named logical unit only from the specified TCP/IP connection or VTAMAPPL.

If you do not enter a TERM, DEV, or LU command following a TCPIP or VTAMAPPL command, the Log Compare Utility processes all devices or logical units for the specified TCP/IP connection or VTAMAPPL. If no resource commands are entered, all logical units found in the log data set are valid for the comparing process.

lu1

Function: *lu1* specifies the name of the logical unit (LU) used in the MASTER log data set.

Format: The value of *lu1* is a 1- to 8-character name that matches the name coded on a WSim network definition LU statement.

-num1

Function: *-num1* specifies a specific half-session of the logical unit being selected for use in the MASTER log data set.

- If *-num1* is not specified when *lu2-num2* are specified, *-num2* is used as the number of the half-session of the LU used in the MASTER log data set.
- If both *-num1* and *lu2-num2* are not specified, the utility uses all half-sessions of the LUs specified for the comparing process.

lu2

Function: *lu2* specifies the name of the LU used in the TEST log data set.

Format: The value of *lu2* is a 1- to 8-character name that matches the name coded on a WSim network definition LU statement.

Default: If *lu2* is not specified, the name specified for *lu1* is used as the name of LU used in the TEST log data set.

-num2
: **Function:** *-num2* specifies a specific half-session of the logical unit being selected for use in the TEST log data set.

- If *-num2* is not specified when *-num1* is specified, *-num1* is used as the number of the half-session of the LU used in the TEST log data set.
- If both *-num1* and *-num2* are not specified, the utility uses all half-sessions of the LUs specified for the comparing process.

MSGTXT command

```
MSGTXT msgtxtid1[,msgtxtid2]
```

The MSGTXT command specifies the name of the message generation deck or STL procedure that the utility selects for the comparing process. The utility selects only those records logged for the specified message generation deck or STL procedure. If you enter this command multiple times for each run, the utility selects multiple message generation decks or STL procedures.

msgtxtid1

Function: The *msgtxtid1* operand specifies the name of the message generation deck or STL procedure used in the MASTER log data set.

Format: The value of the *msgtxtid1* operand is a 1- to 8-character name that matches the name coded on a WSim message generation or STL MSGTXT statement.

msgtxtid2

Function: The *msgtxtid2* operand specifies the name of the message generation deck or STL procedure used in the TEST log data set.

Format: The value of the *msgtxtid2* operand is a 1- to 8-character name that matches the name coded on a WSim message generation or STL MSGTXT statement.

Default: If *msgtxtid2* is not specified, the name specified for *msgtxtid1* is used as the name of the message generation deck or STL procedure used in the TEST log data set.

NTWRK command

```
NTWRK netname1[,netname2]
```

The NTWRK command specifies the name of the network the utility uses for the comparing process.

If you enter more than one NTWRK command, the Log Compare Utility uses the last one entered. If you do not enter a NTWRK command, the Log Compare Utility includes all networks found in the log data set in the comparing process.

netname1

Function: The *netname1* operand specifies the name of the network used in the MASTER log data set.

Format: The value of the *netname1* operand is a 1- to 8-character name that matches the name coded on a WSim network definition NTWRK statement.

netname2

Function: The *netname2* operand specifies the name of the network used in the TEST log data set.

Format: The value of the *netname2* operand is a 1- to 8-character name that matches the name coded on a WSim network definition NTWRK statement.

Default: If *netname2* is not specified, the name used for *netname1* is used as the name of the network used in the TEST log data set.

SELECT command

```
SELECT DATA=(data)
       [,LOC=(row,col)]
       [,SCAN={YES|integer|1}]
```

The SELECT command identifies only the specific records that the utility is to select for the comparing process. This command is mutually exclusive with the EXCLUDE command. The SELECT command is processed after the START command. Thus, the starting display record may be a record that is not selected.

To use this command, you specify a string of data that identifies the record or records to be selected. You can specify the exact position of the data or have the Log Compare Utility scan the entire record.

DATA=(*data*)

Function: The DATA operand specifies the data that identifies the record or records to be selected. The Log Compare Utility requires this operand whenever you specify the SELECT command.

Format: You can represent the data as EBCDIC, hexadecimal (enclosed in single quote marks), or a mixture of both; the data can be up to 20 bytes in length.

Note: SO and SI characters are removed from DBCS data entered and are not counted as part of the previously mentioned 20 bytes. To include an SO or SI character in the data for the data compare, add the hexadecimal equivalent of an SO (X'0E') or SI (X'0F') to the data.

LOC=(*row*,*col*)

Function: The LOC operand indicates the exact panel position where the utility can find the data specified by the DATA operand.

Format: The values for *row* and *col* can be integers from 1 to 255.

Default: The default is 1 for both *row* and *col*.

SCAN={YES|*integer*|1}

Function: The SCAN operand specifies whether or not the utility is to scan the record sequentially for the data specified in the DATA operand. When you specify scanning, the utility searches the record starting at the location specified in the LOC operand. If you omit the LOC operand, the starting location is (1,1) of the screen image.

If you code YES for the SCAN operand, the utility searches the record to the end. If you specify an integer, the utility searches the number of positions specified by *integer*.

Format: *integer* can be an integer from 1 to 32767.

Default: 1.

START command

```
START DATA=(data)
      [,LOC=(row,col)]
      [,SCAN={YES|integer|1}]
```

The START command identifies a specific record in the log data set that the utility will use to start comparing display records. Although you can issue START more than once, subsequent START commands override previous START commands. After the starting display record is selected, the processing for the SELECT and EXCLUDE commands begins. Thus, the starting display record may be a record that is excluded (by way of the EXCLUDE or SELECT command).

DATA=(*data*)

Function: The DATA operand specifies the data that identifies the record the utility uses to start the comparing process. The Log Compare Utility requires this operand whenever you specify the START command.

Format: You can represent the data as EBCDIC, hexadecimal (enclosed in single quote marks), or a mixture of both; the data can be up to 20 bytes in length.

Note: SO and SI characters are removed from DBCS data entered and are not counted as part of the previously mentioned 20 bytes. To include an SO or SI character in the data for the data compare, add the hexadecimal equivalent of an SO (X'0E') or SI (X'0F') to the data.

LOC=(*row*,*col*)

Function: The LOC operand indicates the exact panel position where the utility can find the data specified by the DATA operand.

Format: The values for *row* and *col* can be integers from 1 to 255.

Default: The default is 1 for both *row* and *col*.

SCAN={YES|*integer*|1}

Function: The SCAN operand specifies whether or not the utility is to scan the record sequentially for the data specified in the DATA operand. When you specify scanning, the utility searches the record starting at the location specified in the LOC operand. If you omit the LOC operand, the starting location is (1,1) of the screen image.

If you code YES for the SCAN operand, the utility searches the record to the end. If you specify an integer, the utility searches the number of positions specified by *integer*.

Format: *integer* can be an integer from 1 to 32767.

Default: 1.

SYNCPOINT command

```
SYNCPOINT DATA=(data)
      [,LOC=(row,col)]
      [,SCAN={YES|integer|1}]
```

The SYNCPOINT command identifies specific records to use if the utility synchronizes the log data sets. To format this command, you specify a string of data that identifies the record or records to be selected. You can specify the exact position of the data or have the Log Compare Utility scan the entire record. You can also enter multiple SYNCPOINT commands to define more than one type of record that can be used for a syncpoint. After the Log Compare Utility finds the first syncpoint in the MASTER log data set, it searches for the same syncpoint in the TEST log data set. The Complete Records List will show that syncpoint was met only if it was met in both the MASTER and TEST log data sets.

DATA=(*data*)

Function: The DATA operand specifies the data that identifies the record the utility uses to synchronize the log data sets. The Log Compare Utility requires this operand whenever you specify the SYNCPOINT command.

Format: You can represent the data as EBCDIC, hexadecimal (enclosed in single quote marks), or a mixture of both; the data can be up to 20 bytes in length.

Note: SO and SI characters are removed from DBCS data entered and are not counted as part of the previously mentioned 20 bytes. To include an SO or SI character in the data for the data compare, add the hexadecimal equivalent of an SO (X'0E') or SI (X'0F') to the data.

LOC=(*row*,*col*)

Function: The LOC operand indicates the exact panel position where the utility can find the data specified by the DATA operand.

Format: The values for *row* and *col* can be integers from 1 to 255.

Default: The default is 1 for both *row* and *col*.

SCAN={YES|*integer*|1}

Function: The SCAN operand specifies whether or not the utility is to scan the record sequentially for the data specified in the DATA operand. When you specify scanning, the utility searches the record starting at the location specified in the LOC operand. If you omit the LOC operand, the starting location is (1,1) of the screen image.

If you code YES for the SCAN operand, the utility searches the record to the end. If you specify an integer, the utility searches the number of positions specified by *integer*.

Format: *integer* can be an integer from 1 to 32767.

Default: 1.

TCPIP command

```
TCPIP tcpipid1[,tcpipid2]
```

The TCPIP command specifies the TCP/IP connection ID of the device that the utility uses for the comparing process.

If you enter any TCPIP or VTAMAPPL command, the Log Compare Utility includes only those TCP/IP connections or VTAMAPPLs in the comparing process. If you do not enter any TCPIP or VTAMAPPL commands, TCP/IP connections and VTAMAPPLs found in the log data set are valid for the comparing process.

tcpipid1

Function: The *tcpipid1* operand specifies the name of the TCP/IP connection used in the MASTER log data set.

Format: The value of the *tcpipid1* operand is a 1- to 8-character name that matches the name coded on a WSim network definition TCPIP statement.

tcpipid2

Function: The *tcpipid2* operand specifies the name of the TCP/IP connection used in the TEST log data set.

Format: The value of the *tcpipid2* operand is a 1- to 8-character name that matches the name coded on a WSim network definition TCPIP statement.

Default: If *tcpipid2* is not specified, the name specified for *tcpipid1* is used as the name of the TCP/IP connection used in the TEST log data set.

TERM command

```
TERM termid1[,termid2]
```

The TERM command specifies the name of the terminals that the utility uses for the comparing process. You may enter this command multiple times for each run to process multiple terminals. If you omit this command, the utility considers all devices found in the log data set valid for the comparing process.

If the TERM command does not follow a valid TCPIP or VTAMAPPL command, the Log Compare Utility processes the named terminal on all TCP/IP connections or VTAMAPPLs. If the TERM command does follow a valid TCPIP or VTAMAPPL command, the Log Compare Utility processes the named terminal on the specified TCP/IP connection or VTAMAPPL only.

If you do not enter a TERM, DEV, or LU command following a TCPIP or VTAMAPPL command, the Log Compare Utility processes all devices or logical units for the specified TCP/IP connection or VTAMAPPL. If no resource commands are entered, all terminals found in the log data set are valid for the comparing process.

termid1

Function: The *termid1* operand specifies the name of the device or logical unit (LU) used in the MASTER log data set.

Format: The value of the *termid1* operand is a 1- to 8-character name that matches the name coded on a WSim network definition DEV or LU statement.

termid2

Function: The *termid2* operand specifies the name of the device or LU used in the TEST log data set.

Format: The value of the *termid2* operand is a 1- to 8-character name that matches the name coded on a WSim network definition DEV or LU statement.

Default: If *termid2* is not specified, the name specified for *termid1* is used as the name of the device or LU used in the TEST log data set.

VTAMAPPL command

```
VTAMAPPL vtamapplid1[,vtamapplid2]
```

The VTAMAPPL command specifies the VTAM application ID of the device that the utility uses for the comparing process.

If you enter any TCPIP or VTAMAPPL command, the Log Compare Utility includes only those TCP/IP connections or VTAMAPPLs in the comparing process. If you do not enter any TCPIP or VTAMAPPL commands, all TCP/IP connections and VTAMAPPLs found in the log data set are valid for the comparing process.

vtamapplid1

Function: The *vtamapplid1* operand specifies the name of the VTAMAPPL used in the MASTER log data set.

Format: The value of the *vtamapplid1* operand is a 1- to 8-character name that matches the name coded on a WSim network definition VTAMAPPL statement.

vtamapplid2

Function: The *vtamapplid2* operand specifies the name of the VTAMAPPL used in the TEST log data set.

Format: The value of the *vtamapplid2* operand is a 1- to 8-character name that matches the name coded on a WSim network definition VTAMAPPL statement.

Default: If *vtamapplid2* is not specified, the name specified for *vtamapplid1* is used as the name of the VTAMAPPL used in the TEST log data set.

Process commands

The Log Compare Utility uses process commands to control the comparing process. For example, process commands such as the RUN and END commands define the beginning and end of the comparing process. If you do not code any commands other than RUN and END, the utility compares every display record for all devices or LUs in the log data sets. In addition, process commands determine which reports the utility generates after the run.

ATTRIBUTE command

```
ATTRIBUTE
```

The ATTRIBUTE command specifies that if attributes are present for both records, the utility compares the attributes as well as the screen data. If you do not specify the ATTRIBUTE command, the utility compares only screen data. That is, if attributes exist for both records but you do not use the command, the utility skips the location of the attributes and compares only the screen data, not the attribute values.

If you specify the ATTRIBUTE command and attributes exist for both records, the utility locates and compares the attribute in the MASTER record against the attribute in the TEST record. The utility compares extended attributes, if any, in the same way.

If attributes are not present in one or both of the MASTER and TEST records, the utility treats both screen images like pure data and continues the comparing process. For example, if only the MASTER record has attributes, the utility compares the MASTER record attributes against whatever data appears at the corresponding location in the TEST record and lists records with differences.

Note: The ATTRIBUTE command tells the utility to compare the attributes with the same offsets. Field offsets are compared regardless of the ATTRIBUTE command unless the attribute bytes are masked. For example, an attribute in the master record without a corresponding one at the same offset in the test record always shows as a difference.

CHARATTR command

```
CHARATTR
```

The CHARATTR command specifies that if character attributes are present for both the MASTER and the TEST DSPY record, the Log Compare Utility compares the character attributes along with the that panel data. The Log Compare Utility compares data in fields first, and if no differences are found, it then compares the character attributes.

If you do not specify the CHARATTR command, the Log Compare Utility compares panel data only.

CHECKONLY command

```
CHECKONLY DATA=(data)
          ,CHECKLOC=(row,col,leng)
          [,LOC=(row,col)]
          [,SCAN={YES|integer|1}]
```

The CHECKONLY command specifies that the utility compare only a portion of the records identified for comparison. The DATA, LOC, and SCAN operands specify the string of data the utility uses to identify a record. After the utility finds a record that matches the specifications, the CHECKLOC operand specifies which field is to be compared.

You can issue multiple CHECKONLY commands to define multiple records that require the comparison of only specific fields.

Note: If you have coded the CHECKONLY and the MASK commands, the Log Compare Utility checks all CHECKONLY commands before checking any MASK commands. The Log Compare Utility checks the MASK commands only if the DSPY record did not match any of the CHECKONLY commands. Only one CHECKONLY statement is applied to a screen prior to comparison.

DATA=(*data*)

Function: The DATA operand specifies the data that identifies the record. The Log Compare Utility requires this operand whenever you specify the CHECKONLY command.

Format: You can represent the data as EBCDIC, hexadecimal (enclosed in single quote marks), or a mixture of both; the data can be up to 20 bytes in length.

Note: SO and SI characters are removed from DBCS data entered and are not counted as part of the previously mentioned 20 bytes. To include an SO or SI character in the data for the data compare, add the hexadecimal equivalent of an SO (X'0E') or SI (X'0F') to the data.

CHECKLOC=(*row*,*col*,*leng*)

Function: The CHECKLOC operand identifies the field that the utility compares for the record. The Log Compare Utility requires this operand whenever you specify the CHECKONLY command. You must specify the row and column location of the field and the length of the field (*row,col,leng*).

You can issue the CHECKLOC operand multiple times for a single CHECKONLY command, defining multiple fields to be compared.

If a CHECKLOC operand specifies a location beyond the end of the screen buffer, the Log Compare Utility ignores that location. Therefore, if all of the locations specifies by CHECKLOC operands are out of range, the Log Compare Utility always considers the panels equal because there are no fields specified within the screen buffer to compare.

Format: The values for *row*, *col*, and *leng* are integers from 1 to 255.

LOC=(*row*,*col*)

Function: The LOC operand indicates the exact panel position where the utility can find the data specified by the DATA operand or where the scan is to start if you specify the SCAN operand.

Format: The values for *row* and *col* can be integers from 1 to 255.

Default: The default is 1 for both *row* and *col*.

SCAN={YES|*integer*|<u>1</u>}

Function: The SCAN operand specifies whether or not the utility is to scan the record sequentially for the data specified in the DATA operand. When you specify scanning, the utility searches the record starting at the location specified in the LOC operand. If you omit the LOC operand, the starting location is (1,1) of the screen image.

If you code YES for the SCAN operand, the utility searches the record to the end. If you specify an integer, the utility searches the number of positions specified by *integer*.

Format: *integer* can be an integer from 1 to 32767.

Default: 1.

CURSOR command

```
CURSOR
```

The CURSOR command specifies that the Log Compare Utility compares the cursor position on the MASTER and TEST DSPY records. The Log Compare Utility compares data in fields first. If no difference is detected, the utility then compares the cursor position. MASK and CHECKONLY do not apply to cursor position comparisons.

If you do not specify the CURSOR command, the Log Compare Utility compares panel data only.

END command

```
END
```

The END command specifies that the Log Compare Utility has completed the comparing process; no further processing occurs. The utility ignores any commands entered since the last RUN command.

HEADER command

```
HEADER data
```

The HEADER command defines the data that the utility uses as the page header on the Log Compare Utility output reports. You can code any data for the *data* operand.

data

Function: The *data* operand specifies the header line to be used.

Format: The value of the *data* operand can be up to 27 characters, including blanks and special characters. The report header begins with the first nonblank character coded after the HEADER command and ends with the character in column 71 or after a length of 27 characters, whichever comes first.

Default: If you do not enter this command, the output header defaults to “WSim LOG COMPARE UTILITY”.

MASK command

```
MASK MASKLOC=(row,col,leng)
     [,DATA=(data)]
     [,LOC=(row,col)]
     [,SCAN={YES|integer|1}]
```

The MASK command specifies a specific field or fields that are not to be compared. The DATA, LOC, and SCAN operands specify the string of data the utility uses to identify a specific record. If you code the DATA operand, then the utility masks the field or fields for the records or records identified by the DATA operand. If you omit the DATA operand, then the utility masks the specified field or fields for all records compared. This is referred to as an ***All Mask*** operation in the reports provided by the Log Compare Utility. The Log Compare Utility allows only one All Mask operation during a run.

The LOC and SCAN operands are valid only when you code the DATA operand. The fields or fields to be masked are specified with the MASKLOC operand.

You can issue multiple MASK commands to define more than one record that requires the utility to mask a field.

Note: The Log Compare Utility checks any MASK commands after checking the CHECKONLY commands, if any. If the record does not match any of the CHECKONLY commands, only then does the Log Compare Utility check the MASK commands. If you use the MASKLOC operand and not the DATA operand, the All Mask operation will also be applied. Only one MASK statement (not including "All Mask" operations) is applied to a screen prior to comparison.

MASKLOC=(*row*,*col*,*leng*)

Function: The MASKLOC operand identifies the field that the utility masks for the record. The Log Compare Utility requires this operand whenever you specify the MASK command. You must specify the row and column location of the field and the length of the field (*row,col,leng*).

If a MASKLOC operand specifies a location beyond the end of the screen buffer, the Log Compare Utility ignores that location. Therefore, if all the locations specified by MASKLOC operands are out of range, the Log Compare Utility compares both records in full because no fields within the screen buffer are masked.

You can issue the MASKLOC operand multiple times for a single MASK command, defining multiple fields to be masked. For example, you could mask both date and time, which would be likely to vary for two different simulation runs.

Format: The values for *row*, *col*, and *leng* can be integers from 1 to 255.

DATA=(*data*)

Function: The DATA operand specifies the data that identifies the record. If you omit the DATA operand, then the utility masks the specified field or fields for all records compared. This is referred to as an All Mask operation in the reports provided by the Log Compare Utility.

Format: You can represent the data as EBCDIC, hexadecimal (enclosed in single quote marks), or a mixture of both; the data can be up to 20 bytes in length.

Note: SO and SI characters are removed from DBCS data entered and are not counted as part of the previously mentioned 20 bytes. To include an SO or SI character in the data for the data compare, add the hexadecimal equivalent of an SO (X'0E') or SI (X'0F') to the data.

LOC=(*row*,*col*)
Function: The LOC operand indicates the exact panel position where the utility can find the data specified by the DATA operand or where the scan is to start if you specify the SCAN operand.

Format: The values for *row* and *col* can be integers from 1 to 255.

Default: The default is 1 for both *row* and *col*.

SCAN={YES|*integer*|1}
Function: The SCAN operand specifies whether or not the utility is to scan the record sequentially for the data specified in the DATA operand. When you specify scanning, the utility searches the record starting at the location specified in the LOC operand. If you omit the LOC operand, the starting location is (1,1) of the screen image.

If you code YES for the SCAN operand, the utility searches from the location specified by the LOC operand to the end of the record. If you specify an integer, the utility searches the number of positions specified by *integer*.

Format: *integer* can be an integer from 1 to 32767.

Default: 1.

P command

```
P
```

The P command specifies that the Log Compare Utility stop reading commands input from the operator's terminal and begin reading commands from the SYSIN data set. If you enter this command when the SYSIN data set is not open, the utility continues requesting input from the console. The utility ignores the P command if it finds the command in the SYSIN data set.

REPORT command

```
REPORT [RECORDS]
       [,COMPARES]
       [,DIFFERENCES]
       [,SUMMARY]
```

The REPORT command specifies which reports the utility generates for this run of the Log Compare Utility. If you do not issue a REPORT command, the utility generates only the Active Command List, the Differences Report, and the Summary Report. If you issue a REPORT command without specifying an operand, the utility does not generate any reports except the Active Command List. For detailed information about these reports, refer to “Information you can obtain with the Log Compare Utility” on page 72.

Issue the REPORT command with one or more of the following operands to generate the associated report.

RECORDS
Function: The RECORDS operand specifies that the utility generate a Complete Records List for each device processed during the run.

COMPARES
: **Function:** The COMPARES operand specifies that the utility generate a Compare List for each device processed during the run.

DIFFERENCES
: **Function:** The DIFFERENCES operand specifies that the utility generate a Differences Report for each device processed during the run.

SUMMARY
: **Function:** The SUMMARY operand specifies that the utility generate a Summary Report for the run.

RUN command

```
RUN
```

The RUN command specifies that you have entered all commands and that the utility can begin processing the log data sets. The utility requires this command prior to processing. After processing, the utility resets all variables to their default values before interpreting additional commands. If you issue the RUN command without entering any other commands, the utility processes all records using the default values.

UPPERCASE command

```
UPPERCASE
```

The UPPERCASE command specifies that the utility perform lowercase to uppercase translation before comparing data from the two log data sets. The default for this command is no translation from lower to uppercase; specify the UPPERCASE command to override this default.

* Command

```
* [data]
```

The * command specifies a comment. You can enter any data following the asterisk. Data following the asterisk is listed with input commands as you entered them, but the Log Compare Utility does not include them in the Active Command List or process them during the comparing process.

Chapter 8. Using the Response Time Utility to analyze response times

The Response Time Utility analyzes the log data set and prints reports that list the response times of the system under test. The Response Time Utility calculates response times for each terminal using the log data set record time stamps from a pair of transmit and receive records. Because transmit-receive pairs are determined on a terminal name basis, the terminal names should be unique in a network definition. The Response Time Utility uses a set of default rules for determining the transmit and receive record pairings. However, you can change these rules by specifying operands for the Response Time Utility to use when selecting the transmit and receive records.

Note: WSim provides an online response statistics reporting facility, RSTATS, that is not related to the Response Time Utility. You can use RSTATS to monitor the response times of simulated devices while WSim is executing a script. In contrast, the Response Time Utility is a postprocessor. It uses data from the log data set of a previous WSim simulation run. For more information about the RSTATS facility, refer to *WSim User's Guide*.

The sections in this chapter present information about the following items:

- The types of reports, listings, and graphs you can obtain with the Response Time Utility
- How to run the Response Time Utility, including:
  - Calculation of response times for devices and transactions
  - Estimation of virtual storage
  - Running the Response Time Utility from the WSim/ISPF Interface
  - Execution parameters to use with JCL or a TSO CLIST
  - Examples of JCL and a TSO CLIST for running the Response Time Utility
  - Sample output from an example of an input command file.

Information you can obtain with the Response Time Utility

This section describes the output produced by the Response Time Utility.

Response time reports

Using the REPORT command and the REPORT operand of the VTAMAPPL, APPCLU or TPCIPstatement, you can specify the following levels of response time reports:

TERM	This report provides information about the response times for a particular terminal.
TERMGRP	This report provides cumulative information about the response times for all of the terminals associated with a particular VTAM application, APPC LU or TCP/IP connection.
SUMMARY	This report provides cumulative information about the response times for all of the terminals in a run.

When user transactions have been defined, each of these report types can be requested for each unique transaction type.

The response time report contains the following information:

- The header lines of the report between the dotted lines identify the characteristics of the report. The report contains the following fields in the header lines:

Report type	The title of the report is either TERMINAL REPORT, TERMGRP REPORT, or SUMMARY REPORT.
NETWORK	The name of the network being processed. This field contains the characters ALL NETWORKS if a NTWRK command was not specified.
TCPIP, VTAMAPPL, or APPCLU	The name of the TCP/IP connection, VTAM application, or APPC LU being processed. This field is blank for a Summary Report.
TERMINAL	The name of the terminal being processed. This field is blank for a Termgrp or Summary Report.
PROCESS	Either SYSTEM or ACTUAL.
EXIT	The name of the user exit routine or blank if no exit routine is present.
TERMTYPE	An alphanumeric name describing the type of terminal for this report. This field is blank for a Termgrp or Summary Report.
TIME LIMITS	The time limits for the run as specified by the TIME command.
START TIME	The READY time stamp from the first record found for this report.
END TIME	The READY time stamp from the last record found for this report.
TRANSACTION	The name of the transaction type for this report. This field is omitted if there are no user-defined transactions. The characters "*ALL*" indicate that this report is a summary for all transaction types.

- After the header lines, there is an optional listing of the individual response times identified by the following fields:

RESPONSE TIME	The time between a message being transmitted from WSim and the response being received from the system under test. The format is *hh.mm.ss.th* (hours, minutes, seconds, tenths and hundredths of seconds).
COUNT	The number of times that the corresponding response time was encountered during this run.

- The following fields also appear on the response time report:

MEAN RESPONSE	The total of all the response times divided by the total number of responses.
MEDIAN RESPONSE	The 50th percentile response time. Half of the response times fell above the median and half below.
MODE RESPONSE	The response time that occurred most often in the set of measured response times. The Response Time Utility will print a *mode* value only if there was a unique response time that occurred more often than other response times. Otherwise, "**--**" is printed.

AVERAGE LENGTH	The average length of the messages transmitted and received without regard to the number of response times computed. For SNA devices, this is the length of RUs only.
LOW RESPONSE	The smallest response time value computed during the run.
HIGH RESPONSE	The largest response time value computed during the run.
AVERAGE QUEUE TIME	The average time that a terminal must wait to transmit a message after it is ready to transmit (wait for poll). A single queue time is the difference between the READY and START time stamps of an XMIT record that begins a valid response time. The queue time for a message is not included in the SYSTEM response time, but is a component of the ACTUAL response time.
NUMBER OF RESPONSES	The total number of response times computed for this report.
MESSAGES SENT	The total number of messages transmitted without regard to the number of response times computed. If user transactions have been defined, this number is reported only on the reports that summarize all transaction types. For messages composed of multiple SNA chain elements, each element is counted as a message sent.
MESSAGES RECEIVED	The total number of messages received without regard to the number of response times computed. If user transactions have been defined, this number is reported only on the reports that summarize all transaction types. For messages composed of multiple SNA chain elements, each element is counted as a message received.
PER MINUTE	The number of response times per minute and the number of messages sent and received per minute are reported. These rates are computed only if the time interval for the report (as defined by the START TIME and END TIME values in the header lines) is at least one minute. See "Response time reports" on page 107 for a definition of these fields.
RESPONSES DISCARDED	The number of response times that were not included in any calculations because of virtual storage limitations.
VARIANCE	A measure of the differences in the response times. The sample variance of response times will be computed by the following equation, which yields an unbiased estimate of the true variance: `Variance = [ ss - t²/n ] / (n-1)` where *n* is the total number of responses, *ss* is the sum of squares of the response times, and *t* is the total of the response times. The variance value of "**--**" will be reported if the computed value is greater than 999.9999.

95% CI	Two values computed from the sample response times that delimit, with 95% confidence, the theoretical average response time. The assumption is made that the sample average is approximately normally distributed. This assumption is valid if the total number of responses is at least 25 and the individual response times can be considered to be statistically independent. The limits of the interval are computed by the following formula: $a \pm (1.96 \text{ x } \mathrm{sqrt} [v/n])$ where *a* is the sample average of the response times, *v* is the sample variance, *n* is the total number of responses, and sqrt indicates the square root function. If there are fewer than 25 response times, the mean confidence interval values are not calculated and "**--**" will be printed beside this heading to indicate it is not applicable.
PERCENTILE, RESPONSE TIME, AVERAGE	These titles provide a header line for a table of percentile values. The Percentile column contains the percentile numbers requested by the PERCENT command. The Response Time column contains the computed percentile values. A percentile value is a response time such that the specified percentage of all the response times is less than or equal to this value. A value in the Average column is the average of all response times less than and including the value in the Response Time column.

Transaction record listing

You can use the TPRINT command to request a listing of the log records that are selected for transaction processing by the other input commands such as VTAMAPPL, APPCLU, TCPIP, and TERM. No records are listed unless at least one transaction is defined by a BTRANS command and, optionally, an ETRANS command. One line is printed for each log record selected, and the listing occurs before any other output reports. The following information will be printed for each record:

- TCP/IP connection, VTAM application, or APPC LU name.
- Terminal name.
- Transmit or receive indicator (XMIT or RECV).
- Transaction status.
- Up to 60 data characters from the message in EBCDIC. Unprintable characters will be translated to periods (.)
- User data byte from the log record header (printed in hexadecimal and EBCDIC)
- START, STOP, and READY time stamps.

The transaction status can be either B-*type* or E-*type* to indicate whether the record begins or ends a transaction. The *type* is the name from the BTRANS TYPE operand. There is one exception; if you define a transaction with a BTRANS command only (no ETRANS command), and you specify PROCESS ACTUAL, the ending status E-*type* is not printed. The status area will also be blank for a record that does not begin or end a transaction. In the event that MATCH=LAST has been specified on a BTRANS or ETRANS command, it is possible that the status fields of several successive records might have B-*type* or E-*type* printed. If this is the case, only the last of these records is considered the actual beginning or ending of the transaction.

Response listing file

The Response Time Utility can create a sequential data set consisting of one record for each response time computed during the run. The basic record contains the following information:

Columns	Field Contents
1-8	TCP/IP connection, VTAM application, or APPC LU name (EBCDIC)
10-22	Terminal name
10-17	Name
18-22	Session number
24-31	Transaction name
34	User data byte (hexadecimal)
37-40	Transaction start time (binary hundredths of seconds)
44-51	Transaction start time (EBCDIC)
54-57	Response time (binary hundredths of seconds)
61-68	Response time (EBCDIC)
71-78	Message generation deck name

The extended record contains the following additional information:

Columns	Field Contents
81-88	Response time in seconds, long floating point format
90-98	Response time in EBCDIC, *nnnnnn.nn* format
100-107	Time of day in binary when the transaction end log record was written to the log data set
100-103	Time of day in binary 1/100 seconds
104-107	Date in *00yydddf* format
109-116	Time of day in EBCDIC when the transaction end log record was written to the log data set, in *hhmmssth* format
118-122	Date in EBCDIC when the transaction end log record was written to the log data set, in *yyddd* format
124-131	Network name
133-159	HEADER command character string.

The JCL *ddname* for the file must be LISTDD. The LISTDD DD and the LIST or LISTX execution parameter must be specified for the run before the Response Time Utility will attempt to create the response listing file. If transaction processing is not being done, the transaction name field will be all blanks, and the transaction start time will be a READY time stamp for PROCESS ACTUAL or a STOP time stamp for PROCESS SYSTEM. If transactions have been specified, the transaction start time on each record will be the time stamp specified by the BTRANS statement TIME operand. Each record in the LISTDD file will be padded with blanks to a length of 80 bytes for the LIST execution parameter, 160 bytes for LISTX. The default BLKSIZE for the LISTDD statement is 80 bytes for the LIST execution parameter, 160 bytes for LISTX.

The extended records produced when the LISTX execution parameter is specified provide additional information that is required for record processing by program products such as Service Level Reporter (SLR) using user-defined SLR Adaptable Log Layout (ALL) log tables and user programs.

Response Time Frequency Distribution

The Response Time Frequency Distribution is a histogram showing the percentage of occurrence of each calculated response time for the period of time defined for the run. Figure 31 on page 125 shows an example of the histogram. The y-axis (vertical) shows the percentage of responses in 2 percent increments from 0 to 100. The x-axis (horizontal) shows the response time values in even tenths of seconds using 50 increments. The first response time value used on the x-axis is the lowest response time truncated to an even tenth of a second. For example, if the lowest response time is 2.36 seconds, the first response time value used on the x-axis is 2.2 seconds.

You can use the GRAPH command to specify the increment value to be used on the x-axis (in tenths of seconds). If a GRAPH number of 0 is specified, the increment value is determined by subtracting the lowest response time from the highest response time and dividing the difference by 50.

You can control when the frequency distribution graphs are printed by using the REPORT command and the REPORT operand on the VTAMAPPL, APPCLU, TCPIP, and TERM commands.

Cumulative Response Time Distribution

The Cumulative Response Time Distribution is a graph showing the cumulative distribution of response times for the period of time defined for the run. Figure 32 on page 126 shows an example of this graph. The y-axis (vertical) shows the percentage of responses in 2 percent increments from 0 to 100. The x-axis (horizontal) shows the response time values in even tenths of seconds using 50 increments. The first response time value used on the x-axis is the lowest response time truncated to a full second. For example, if the lowest response time is 2.36 seconds, the first response time value used on the x-axis is 2.2 seconds.

You can use the CGRAPH command to specify the increment value to be used on the x-axis (in tenths of seconds). If a CGRAPH number of 0 is specified, the increment value is determined by subtracting the lowest response time from the highest response time and dividing the difference by 50.

You can control when the cumulative graphs are printed by using the REPORT command and the REPORT operand on the VTAMAPPL, APPCLU, TCPIP, and TERM commands.

Time Graph of Responses

A Time Graph of Responses shows how the response times for a terminal or group of terminals vary over a period of time. Figure 33 on page 127 is an example of this graph. Each horizontal line of a time graph represents a time interval of user-specified length and plots the minimum, average, and maximum response times computed during that interval. Since response times are computed when WSim receives records, it is possible that an individual response time period may begin in one TGRAPH interval and end in a later TGRAPH interval. In this case, the response time is accumulated in the later interval. To minimize possible distortion of the time graph, make sure that the chosen INTERVAL value on the TGRAPH command exceeds the maximum response time for each interval.

Each response time is rounded to the nearest increment value as specified by the INCR operand. You can set the range of response times plotted by specifying the ORIGIN and INCR operands. If a response time is outside the specified range, then its character will be plotted at the limit of the range. If only one response time is accumulated during a single interval, then the average symbol will be used to plot the response time. For example:

```
TIME 0900-0905
REPORT TERM=(TGRAPH)
TGRAPH INTERVAL=10,ORIGIN=5,INCR=1
RUN
END
```

The time graph routine will accumulate the minimum, average, and maximum response times over 30 ten-second intervals. The range of the plotted response times will be from 5 to 15 seconds in tenths-of-second increments.

You can control when the time graphs are printed by using the REPORT command and the REPORT operand on the VTAMAPPL, APPCLU, TCPIP, or TERM commands.

Running the Response Time Utility

The Response Time Utility requires information about how you want to calculate the response time statistics. You supply this information by using control commands entered from an input file or the console. The JCL or TSO CLIST you use to run the Response Time Utility provides the following locations:

- Log data set
- Control command input file
- Response Time Utility program
- The printer.

The following sections describe more information about the Response Time Utility:

- Response time calculations for devices
- Response time calculations for transactions
- Virtual storage estimation
- Execution parameters
- Examples of JCL and a TSO CLIST
- Output generated by a sample control command file

Note: The Response Time Utility allows user routines to gain control and perform any desired functions using the log records with the EXIT command. If you do not code a user exit with the EXIT command, the WSim-supplied exit routine processes the log records and calculates the values for the standard output reports according to the previously defined rules. If you do code a user exit with the EXIT command, it receives control for each log record that satisfies the input command specifications. For more information about user exit routines for the Response Time Utility, refer to *WSim User Exits*.

Calculating response times for terminals

The Response Time Utility calculates response times for each terminal by computing the difference between a time stamp from a record transmitted by the terminal and a time stamp from a record received by the terminal. Three time stamps are associated with each record and are identified as START, STOP, and READY. For information about how these time stamps are set, see "How messages are time stamped" on page 164.

Rules for calculating response times

The rules for calculating a response time are determined by one of the following input commands:

PROCESS SYSTEM
The response time is the difference between the time when WSim sent the last byte of a transmission and when WSim received the first byte of data from the system under test (receive record START time stamp minus transmit record STOP time stamp). If multiple records are transmitted without receiving a record, the last transmit record is used in making the response calculation. Specifying PROCESS=SYSTEM is the equivalent of specifying BTRANS with MATCH=LAST, RECORD=XMIT, and TIME=STOP; and ETRANS with MATCH=FIRST, RECORD=RECV, and TIME=START.

PROCESS ACTUAL
For nonbuffered terminals, the response time is the difference between the time WSim is able to send a message and the time that the last data character is received by the WSim host processor (receive record READY time stamp minus the transmit record READY time stamp). The receive record used is the last message received by the terminal before the next transmit message.

For SNA devices, the receive READY time stamp is taken from the record that is the last-in-chain RU of the last receive chain before the next transmit chain. For other buffered terminals, the receive READY time stamp is taken from the last receive record before the next transmit record. Specifying PROCESS=ACTUAL is the equivalent of specifying BTRANS with MATCH=FIRST, RECORD=XMIT, and TIME=READY; and ETRANS with MATCH=LAST, RECORD=RECV, and TIME=READY.

Refer to “Calculating response times for transactions” on page 115 for information about how the Response Time Utility calculates response times for user-defined transactions.

Response time accuracy

All response times will be computed to the accuracy afforded by the clocks from which the WSim time stamps are obtained.

Note: You can use the TRUNC command to specify that all time stamps are to be truncated to tenths of seconds before performing any calculations.

Device dependencies

The messages used to compute response times depend on the following devices:

SNA Terminals
By default, all SNA control commands, such as data flow control, network control, and session control, and all SNA responses will be discarded by the Response Time Utility. Only request units of the function management class (FM data RUs) will be considered as valid messages for computing response times. It is possible, however, to override this default if transaction processing is being performed (at least one BTRANS command is specified). For more information, refer to the SNA operand descriptions on the BTRANS and ETRANS commands.

3270 Terminals
For 3270 terminals, you can use the UNLOCK command to specify whether or not 3270 keyboard unlock messages are to be considered as valid data messages. For basic support 3270 devices, a keyboard unlock

message is defined to be a 2-byte message (discounting the BSC escape character) consisting of a write command (X'F1' or X'F5') followed by a write control character (WCC) with the keyboard restore bit on. For 3270 devices with extended function support, a keyboard unlock message can be contained within an outbound 3270 data stream Write Structured Field.

Note: The UNLOCK command is ignored when doing transaction processing.

5250 Display Devices
A response time for a 5250 display device will begin when the terminal transmits the first RU of a chain. The response time will end when the last RU of a chain is received, provided that a write to display (WTD) command (X'0411') that unlocks the keyboard appears somewhere in the chain. Each RU of the received chain will be scanned for the WTD command, but the scan will not detect a WTD command that is split between two RUs. These rules are followed only if PROCESS ACTUAL is specified and user-defined transactions are not specified.

Calculating response times for transactions

Using the Response Time Utility transaction processing, you can specify the messages that delimit logical transactions for computing response times. A transaction for a terminal is defined to be a user-specified pair of delimiting records, where any number of messages can be transmitted or received between the delimiting records. You can designate a transaction to begin or end with any record transmitted or received by a terminal. Using transaction definitions, the following types of times can be measured:

- Terminal response time (XMIT --> RECV)
- WSim processing time (RECV --> XMIT)
- Delay time (XMIT --> XMIT, RECV --> RECV)
- Event interval time (any combination).

Transaction processing

Transaction processing is indicated to the Response Time Utility by specifying BTRANS (Begin Transaction) commands and, optionally, ETRANS (End Transaction) commands. The BTRANS command defines the beginning of a transaction for a terminal and ETRANS defines the end. When you define transactions to the Response Time Utility, you must use a BTRANS command, which can be followed by one or more ETRANS commands to define the possible endings. The BTRANS and ETRANS commands used to define a single transaction can be separated in the input data set by other valid Response Time Utility commands. If a BTRANS command is not followed by an ETRANS command (for example, another BTRANS or a RUN command is encountered before an ETRANS command), the Response Time Utility automatically ends the transaction response time according to the option specified on the PROCESS command.

As with other commands, the BTRANS and ETRANS commands are valid only during the processing run in which they are specified. The maximum number of transaction types that can be specified in any one run is defined by the TRAN parameter on the Response Time Utility EXEC statement, with a default of 10. For a single processing run, response times are accumulated according to transaction types. Each terminal, terminal group, or summary RESPBLK contains a pointer to a chain of RESPBLKs, one for each of the transaction types. The response times are recorded individually by transaction type. The WSim output reports reference the transaction types by name, as specified by the TYPE operand on the BTRANS command. If multiple BTRANS commands specify the same name for the TYPE operand, the response times for all of those transactions will be grouped into a single report.

If a syntax error occurs on a BTRANS or ETRANS command, that entire transaction definition is discarded. The Response Time Utility then sets up to begin a new specification (for example, a BTRANS command is expected next after an error).

Note: The UNLOCK command is ignored when doing transaction processing.

***The order of BTRANS and ETRANS commands*:** The order in which the BTRANS and ETRANS pairs are specified is important. The Response Time Utility locates the beginning record of a transaction for a terminal by comparing transmitted and received records with the BTRANS specifications in the order in which the BTRANS commands appeared in the input data set. If a message could satisfy more than one BTRANS specification, the transaction type is determined by the first BTRANS command entered. Similarly, if a message could satisfy more than one ETRANS specification, the record is taken to be the end of the transaction type specified by the first ETRANS command entered, provided that the corresponding BTRANS specification was previously satisfied.

All specified transaction types are considered for each terminal in the analysis. Once a record has been found that satisfies a BTRANS specification for a terminal, all subsequent qualifying records for that terminal are ignored for that particular BTRANS specification until a record is found that satisfies a corresponding ETRANS specification. However, all qualifying records continue to be considered for beginning other available transaction types for the terminal.

***BTRANS and ETRANS operands*:** The BTRANS and ETRANS TEXT operands specify characters to be compared with data in the data portions only of the log records. The header of a log record is not considered during the data scan function of transaction processing.

Response times are computed using the time stamps specified by the BTRANS and ETRANS TIME operands. If transactions are specified by BTRANS commands only, the response times are calculated from the time stamp specified by the BTRANS TIME operand to the receive time stamp specified by the nontransaction rules (PROCESS SYSTEM or ACTUAL).

Transaction processing examples

The following examples explain some of the details of transaction processing.

Example 1:

```
TERM TR3270
BTRANS TEXT=(LOGON),SCAN=YES,TYPE=LOGON,TIME=STOP,
       RECORD=XMIT
ETRANS TEXT=(READY),LENG=5,LOC=6,SCAN=YES
BTRANS LOG=C1,TYPE=LOGBYTE
ETRANS LOG=B
RUN
END
```

The first BTRANS command searches transmit records for the characters LOGON. The entire data portion of each transmit log record is scanned. When a match is found, the transmit STOP time stamp is saved. Next, receive records are scanned for the characters READY. Only receive records are scanned because the RECORD operand value defaults to RECV for ETRANS specifications. Only records having at least five data characters are considered; the scan begins with the sixth data character and continues to the end of the record. When a match is found, the READY time stamp is used with the saved transmit time stamp to compute a response time.

If a transmit record does not contain the characters LOGON or a LOGON type of transaction has been started but not ended, the user data byte in the log record header is compared with the character A (X'C1'). If a match occurs, the record is taken as the beginning record of this second transaction type. Subsequent receive records are inspected for the character B (X'C2') in the user data byte of the log record header.

The response times of both transaction types are accumulated and reported separately for the terminal TR3270.

Example 2:

```
BTRANS LOC=3,SCAN=25,TEXT=(TIME),
       TYPE=00000001,RECORD=RECV
ETRANS LOG=FF,TEXT=(READY)
RUN
END
```

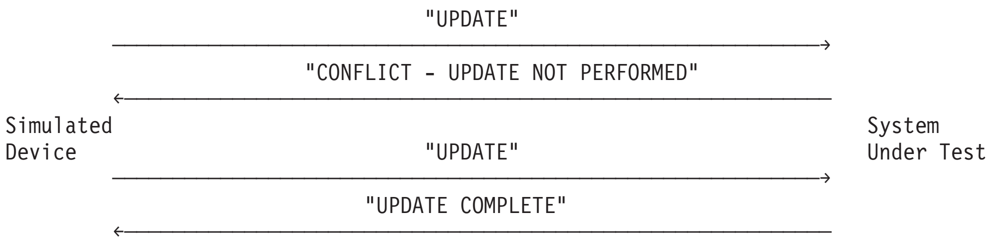
The responses for the transaction type defined by the BTRANS/ETRANS pair are accumulated for all terminals represented on the log data set. The BTRANS command causes a scan of all receive records for the characters TIME beginning with the third data byte of the log record. A maximum of 25 comparisons of length four will be performed on the record, with each comparison beginning at the next consecutive byte. After the beginning transaction record has been found, the ETRANS specifications are used to locate the ending transaction record. If a receive record has X'FF' in the user-data byte of the log header, the record is taken as the end of the transaction regardless of the data portion of the record. If the user data byte is not X'FF', a single comparison for the characters READY is made starting with the first data byte in the record.

Example 3:

```
BTRANS TEXT=(LOGON),TYPE=LOGON
BTRANS TEXT=(EDIT),TYPE=EDIT
BTRANS TEXT=(TOP),TYPE=EDIT
BTRANS TEXT=(SAVE),TYPE=EDIT
BTRANS TEXT=(TIME),TYPE=TIME
ETRANS TEXT=(READY),SCAN=YES
BTRANS TEXT=(LOGOFF),TYPE=LOGOFF
T      TR3270,REPORT=(LIST,TRANS,TGRAPH)
RUN
END
```

The defined transaction types are processed for the terminal TR3270. For this terminal, a listing of the log records, a response time report for each transaction type, and a time graph will be printed. A single response time begins when the terminal transmits the characters specified in one of the BTRANS commands. The TYPE=TIME transaction response time is ended when the characters READY are received. Any other transaction type response time is ended when the terminal receives any message. The response times computed according to the BTRANS commands with TYPE=EDIT will be accumulated in a single output report.

Example 4: Consider the following sequence of transmissions:



The following BTRANS and ETRANS pair causes a transaction to begin when the first UPDATE is transmitted by the simulated device and end when the UPDATE COMPLETE message is received.

```
BTRANS   TYPE=SUCCESS,TEXT=(UPDATE),MATCH=FIRST
ETRANS   TEXT=(UPDATE COMPLETE)
```

Contrast the above specification with the following BTRANS and ETRANS pair. With these commands, the transaction SUCCESS begins with the second UPDATE transmitted by the simulated device and ends when the UPDATE COMPLETE message is received.

```
BTRANS   TYPE=SUCCESS,TEXT=(UPDATE),MATCH=LAST
ETRANS   TEXT=(UPDATE COMPLETE)
```

Example 5:

```
BTRANS TEXT=(UPDATE),TYPE=UPDATE
ETRANS TEXT=(UPDATE COMPLETE)
ETRANS TEXT=(CONFLICT)
RUN
END
```

This example uses multiple ETRANS commands for a single BTRANS command. The responses for the transaction defined by these commands accumulates for all terminals represented on the log data set. A transmitted record containing the characters UPDATE signals the beginning of transaction UPDATE for a terminal. The receipt of either a record containing the characters UPDATE COMPLETE or a record containing the characters CONFLICT signals the end of the transaction. Therefore, for the sequence of transmissions above, the transaction UPDATE begins with the first UPDATE transmitted by the simulated device and ends when the CONFLICT - UPDATE NOT PERFORMED message is received.

Estimating virtual storage

The amount of storage necessary to run the Response Time Utility varies considerably, depending on the number of lines and terminals included, the amount of transaction processing performed (specification of BTRANS and ETRANS commands), the number of graphs generated, the length of the run being analyzed, and the message traffic rates for the run. However, you should have ample storage if you run the Response Time Utility in the same region size that you used to run WSim. If for some reason this is not possible, the following recommendations will help to minimize the storage requirements:

1. Make several smaller runs of the Response Time Utility, instead of one large run. Specify a subset of the lines and terminals to be included in each.
2. Specify the smallest possible value for the TRAN execution parameter.
3. Specify a small value for the RESP execution parameter.
4. Use several runs to generate the graphs.
5. Run the Response Time Utility only for the time interval in which you are interested.

Using the WSim/ISPF Interface

You can run the Response Time Utility from the WSim/ISPF Interface. To do this, follow these steps:

1. Invoke the WSim/ISPF Interface main panel from ISPF. The method you use to do this depends on how the WSim/ISPF Interface application is installed at your site. If you are not sure how to do this, see your system programmer for assistance.
2. Select option 8 from the WSim/ISPF Interface main panel and press **Enter**. The Analyze Response Times panel is displayed.

   Note: You can also type "RESPONSE" on the command line and press **Enter** to display this panel.
3. Fill in the appropriate information on this panel and press **Enter** to run the Response Time Utility.

For more information on the WSim/ISPF Interface, see Chapter 2, "Running WSim with the WSim/ISPF Interface," on page 5.

Using Response Time Utility execution parameters

You can enter the following parameters in the PARM field of the JCL EXEC statement or the TSO CLIST CALL statement when you run the Response Time Utility.

CONSOLE
Specifies that a WTOR is issued to the WSim operator console for the input control commands. If CONSOLE is not specified, the control commands are read from the SYSIN data set.

LIST
Specifies the inclusion of the response listing file. Refer to "Response listing file" on page 111 for information about the response listing file.

LISTX
Specifies the inclusion of the extended response listing file that can be processed by the SLR using user-defined SLR Adaptable Log Layout (ALL) log tables. Refer to "Response listing file" on page 111 for information about the response listing file.

NOLIST
Specifies the exclusion of the response listing file. NOLIST is the default.

PRTLNCNT=*nnn*
Specifies the maximum number of lines to be printed on a page of output before ejecting to a new page where *nnn* is an integer from 35 to 255. The default value for PRTLNCNT is 60.

RESP=*integer*
Specifies the initial number of unique responses that can be saved in a single response time table where *integer* is from 1 to 9999 with a default value of 50. The value you specify for RESP will be used to allocate the first response time table for each terminal, for each line or node, and for the summary table. The size of a response time table will be dynamically increased in increments of 100 unique response times (800 bytes) as needed during the run. To minimize storage requirements, you should code a value for RESP that is equal to the least number of expected response times for any terminal in the analysis.

ROUTCDE=(*n***,***n***,...)**
Specifies the message routing codes to be used in writing Response Time

Utility messages to the operator where each *n* specifies a system routing code that defines a console destination for all WTOs and WTORs written by the Response Time Utility. Each *n* is an integer from 1 to 16. The default ROUTCDE is 8.

TRAN=*integer*
Specifies the maximum number of response transaction types allowed for each run, where *integer* is from 1 to 999 with a default value of 10. One transaction type will be defined by a pair of BTRANS and ETRANS statements.

Using JCL

The following JCL statements are required to run the Response Time Utility on MVS.

Statement	Function
JOB	Initiates the job.
JOBLIB DD	Defines the data set containing the WSim host processor modules.
EXEC	Specifies the program name.
SYSPRINT DD	Defines the output printer.
LISTDD DD	Defines the response listing file (optional).
SYSUT1 DD	Defines the log data set input file. The SYSUT1 DD statement contains an optional BLKSIZE parameter defining the maximum block size for the input data. If you specify this value, it should agree with the BLKSIZE parameter on the LOGDD DD statement from the WSim execution JCL that created the data set. If you do not specify a BLKSIZE parameter, the value is taken from the LOGDD data set if: 1. It is on a labelled tape and it is not overridden with the JCL (by way of ALLOC) 2. It is on a disk data set and it is not overridden with the JCL (by way of ALLOC).
SYSIN DD	Defines the control command input file.

If a user exit routine is specified with the EXIT command, concatenate the library containing the user exit and the WSim load module library on the JOBLIB DD statement.

The following example shows the JCL that you can use to run the Response Time Utility when the log data set is on tape.

```
//RESPJOB1 JOB
//JOBLIB   DD   DSN=WSIM.SITPLOAD,DISP=SHR
//         DD   DSN=USER.LOAD,DISP=SHR
//GO       EXEC PGM=ITPRESP,PARM='RESP=12,TRAN=14,LIST'
//SYSPRINT DD   SYSOUT=A
//LISTDD   DD   DSN=RESPLIST,UNIT=3380,VOL=SER=USERPK,
//              DISP=(,KEEP),SPACE=(CYL,(2)),
//              DCB=(LRECL=80,BLKSIZE=80)
//*             MESSAGE LOGGING INPUT DATA SET ON TAPE
//SYSUT1   DD   UNIT=TAPE,VOL=SER=LOGTAP,LABEL=(,NL),DISP=OLD
//SYSIN    DD   *
   VTAMAPPL VAPPL1
   TERM APPL1
   TERM APPL2
   BTRANS TEXT=(TIME),SCAN=YES,TYPE=TIME
   ETRANS TEXT=(READY),SCAN=YES
   BTRANS LOG=01,TYPE=LOGBYTE
   ETRANS LOG=02
   TIME 0039-0041
   RUN
   PROCESS ACTUAL
   TIME 0044-0048
   EXIT EXITMOD
   RUN
   END
/*
```

The following example shows the JCL that you can use when the log data set is on disk.

```
//RESPJOB2 JOB
//JOBLIB   DD   DSN=WSIM.SITPLOAD,DISP=SHR
//         DD   DSN=USER.LOAD,DISP=SHR
//GO       EXEC PGM=ITPRESP,PARM='RESP=12,TRAN=14,LIST'
//SYSPRINT DD   SYSOUT=A
//LISTDD   DD   DSN=RESPLIST,UNIT=3380,VOL=SER=USERPK,
//              DISP=(,KEEP),SPACE=(CYL,(2)),
//              DCB=(LRECL=160,BLKSIZE=160)
//*             MESSAGE LOGGING INPUT DATA SET ON DISK
//SYSUT1   DD   DSN=WSIM.MSGLOG,DISP=SHR
//SYSIN    DD   *
   VTAMAPPL VAPPL1
   TERM APPL1
   TERM APPL2
   VTAMAPPL VAPPL2
   BTRANS TEXT=(TIME),SCAN=YES,TYPE=TIME
   ETRANS TEXT=(READY),SCAN=YES
   BTRANS LOG=01,TYPE=LOGBYTE
   ETRANS LOG=02
   TIME 0039-0041
   RUN
   PROCESS ACTUAL
   TIME 0044-0048
   RUN
   END
/*
```

In the preceding example, each RESPBLK is initially allocated with space to accumulate 10 unique response times. This space allocation will be dynamically increased if more than 10 unique response times are computed. Each set of input commands can contain a maximum of two transaction type specifications.

Two passes are made through the log data set. On the first pass, the analysis is made for the two VTAMAPPLs, with APPL1 and APPL2 being the only logical units processed for line VTAMAPPL VAPPL1. Response times are calculated according to the two transaction types specified by the BTRANS and ETRANS pairs. The second pass uses the second set of time limits without rewinding the tape, and processes all terminals and devices on lines 010022 and 010023.

Using a TSO CLIST

The following example shows a CLIST to run the Response Time Utility under TSO.

```
ALLOC DDNAME(SYSPRINT) SYSOUT(A)
ALLOC DDNAME(SYSUT1) DSNAME('WSIM.LOGDATA') SHR
ALLOC DDNAME(SYSIN) DSNAME('USER.RSPCMNDS.DATA')
CALL  'WSIM.SITPLOAD(ITPRESP)' 'RESP=50'
FREE DDNAME(SYSPRINT)
FREE DDNAME(SYSUT1)
FREE DDNAME(SYSIN)
```

Understanding sample output

The following pages contain some examples of the output created when you use the WSim/ISPF Interface, JCL or a TSO CLIST to run the Response Time Utility with the input command file shown in Figure 28. Figure 29 on page 123 through Figure 33 on page 127 show the examples.

```
*
*  SPECIFY RUN PARAMETERS
*
HEADER   TRANSACTION TEST
TIME     ALL
REPORT   LEVEL=TERM,
         TERM=(LIST,GRAPH,CGRAPH,TGRAPH,TRANS),
         SUMMARY=(NOLIST,GRAPH,CGRAPH,TGRAPH,NOTRANS)
TPRINT   YES
TRUNC    YES
TGRAPH   ORIGIN=0,INCR=1,INTERVAL=30
GRAPH    2
CGRAPH   2
PERCENT  10,20,30,40,50,60,70,80,90,95
*
*  DEFINE TRANSACTION
*
BTRANS   TYPE=ONLY,
         TEXT=(SECONDARY MESSAGE),
         MATCH=FIRST,
         LENG=1,
         SCAN=YES
ETRANS   TEXT=(PRIMARY MESSAGE),
         SCAN=YES
*
*  DEFINE RESOURCES TO BE SELECTED AND
*  THEIR OUTPUT OPTIONS
*
NTWRK    SAMPNET
 VTAMAPPL VAPPL1
  TERM   DEVLU2X
 VTAMAPPL VAPPL2
  TERM   DEVLU2
*
RUN
END
```

Figure 28. Example of input commands for the Response Time Utility

```
-----------------------------------------------------------------------------------------------------------------------------
TERMINAL REPORT   NETWORK  SAMPNET           PROCESS  SYSTEM      TIME LIMITS ALL             TRANSACTION ONLY
                  VTAMAPPL VAPPL1            EXIT                 START TIME  094105
                  TERMINAL DEVLU2X           TERMTYPE LU2         END TIME    100448
-----------------------------------------------------------------------------------------------------------------------------

RESPONSE TIME    COUNT    RESPONSE TIME   COUNT   RESPONSE TIME    COUNT   RESPONSE TIME    COUNT   RESPONSE TIME    COUNT
         0.00      25              0.10     27             0.20     27              0.30     13              0.40     10
         0.50       8              0.60      3             0.70      1              0.80      6              0.90      7
         1.00       3              1.10      5             1.20      4              1.30      6              1.40     10
         1.50       5              1.60      4             1.70      2              1.80      1              1.90      1
         2.00       1              2.10      5             2.20      2              2.30      3              2.40      1
         2.50       1              2.60      4             2.70      4              2.80      3              2.90      6
         3.00       6              3.10      4             3.20      3              3.30      3              3.40      5
         3.50       3              3.60      7             3.70      4              3.80      4              3.90      2
         4.00       4              4.10      5             4.20      1              4.30      2              4.40      3
         4.50       2              4.60      3             4.70      1              4.80      4              4.90      3
         5.00       5              5.10      2             5.20      4              5.30      2              5.40      5
         5.50       1              5.60      2             5.70      1              5.80      1              5.90      1
         6.00       1              6.20      2             6.30      1              6.40      2              7.30      1
         8.40       1              9.30      1            20.90      1

MEAN RESPONSE              2.12    MESSAGES SENT         304    NUMBER OF RESPONSES        301
MEDIAN RESPONSE            1.40       AVERAGE LENGTH   1,768       PER MINUTE               12
MODE RESPONSE                --       PER MINUTE          13    RESPONSES DISCARDED          0
LOW RESPONSE               0.00    MESSAGES RECEIVED     433    VARIANCE                5.0773
HIGH RESPONSE             20.90       AVERAGE LENGTH     400    95 PERCENT CI       (1.87,2.38)
AVERAGE QUEUE TIME         0.24       PER MINUTE          19


PERCENTILE    RESPONSE TIME         AVERAGE
    10                0.10             0.01
    20                0.20             0.07
    30                0.30             0.12
    40                0.80             0.22
    50                1.40             0.41
    60                2.50             0.65
    70                3.20             0.97
    80                4.00             1.30
    90                5.00             1.65
    95                5.50             1.84
```

Figure 29. The Response Time Utility Terminal Report

```
                                             LISTING OF TRANSACTION RECORDS

APPCLU/TCPIP/    DEV/LU/TP       TYPE  STATUS  LOG RECORD DATA                                     USER     START    STOP     READY
VTAMAPPL            NAME                                                                           DATA     TIME     TIME     TIME
    NAME
  VAPPL2         DEVLU2          RECV          5CSTART SENDING                                     00     09405770 09405770 09405770
  VAPPL2         DEVLU2          XMIT  B-ONLY  'CH00001 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09405790 09405790 09405770
  VAPPL2         DEVLU2          RECV  E-ONLY  5C00001 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09410150 09410150 09410150
  VAPPL2         DEVLU2          XMIT  B-ONLY  'CH00002 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09410320 09410320 09410300
  VAPPL2         DEVLU2          RECV  E-ONLY  5C00002 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09410850 09410850 09410850
  VAPPL1         DEVLU2X         RECV          5CSTART SENDING                                     00     09411250 09411250 09411250
  VAPPL2         DEVLU2          XMIT  B-ONLY  'CH00003 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09411300 09411300 09411260
  VAPPL1         DEVLU2X         XMIT  B-ONLY  'CH00001 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09411300 09411300 09411260
  VAPPL1         DEVLU2X         RECV  E-ONLY  5C00001 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09411820 09411820 09411820
  VAPPL2         DEVLU2          RECV  E-ONLY  5C00003 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09411820 09411820 09411820
  VAPPL1         DEVLU2X         XMIT  B-ONLY  'CH00002 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09411980 09411980 09411980
  VAPPL2         DEVLU2          XMIT  B-ONLY  'CH00004 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09411980 09411980 09411980
  VAPPL1         DEVLU2X         RECV  E-ONLY  5C00002 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09411980 09411980 09411980
  VAPPL1         DEVLU2X         XMIT  B-ONLY  'CH00003 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09412160 09412160 09412150
  VAPPL1         DEVLU2X         RECV  E-ONLY  5C00003 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09412180 09412180 09412180
  VAPPL1         DEVLU2X         XMIT  B-ONLY  'CH00004 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09412390 09412390 09412380
  VAPPL2         DEVLU2          RECV  E-ONLY  5C00004 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09412390 09412390 09412390
  VAPPL1         DEVLU2X         RECV  E-ONLY  5C00004 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09412400 09412400 09412400
  VAPPL1         DEVLU2X         XMIT  B-ONLY  'CH00005 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09412580 09412580 09412560
  VAPPL2         DEVLU2          XMIT  B-ONLY  'CH00005 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09412580 09412580 09412560
  VAPPL2         DEVLU2          RECV  E-ONLY  5C00005 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09412580 09412580 09412580
  VAPPL1         DEVLU2X         RECV  E-ONLY  5C00005 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09412580 09412580 09412580
  VAPPL2         DEVLU2          XMIT  B-ONLY  'CH00006 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09412840 09412840 09412840
  VAPPL1         DEVLU2X         XMIT  B-ONLY  'CH00006 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09412840 09412840 09412820
  VAPPL2         DEVLU2          RECV  E-ONLY  5C00006 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09412840 09412840 09412840
  VAPPL1         DEVLU2X         RECV  E-ONLY  5C00006 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09413090 09413090 09413090
  VAPPL2         DEVLU2          XMIT  B-ONLY  'CH00007 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09413090 09413090 09413090
  VAPPL2         DEVLU2          RECV  E-ONLY  5C00007 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09413100 09413100 09413100
  VAPPL1         DEVLU2X         XMIT  B-ONLY  'CH00007 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09413230 09413230 09413230
  VAPPL2         DEVLU2          XMIT  B-ONLY  'CH00008 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09413230 09413230 09413230
  VAPPL1         DEVLU2X         RECV  E-ONLY  5C00007 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09413240 09413240 09413240
  VAPPL1         DEVLU2X         XMIT  B-ONLY  'CH00008 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09413600 09413600 09413600
  VAPPL2         DEVLU2          RECV  E-ONLY  5C00008 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09413600 09413600 09413600
  VAPPL1         DEVLU2X         RECV  E-ONLY  5C00008 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09413730 09413730 09413730
  VAPPL2         DEVLU2          XMIT  B-ONLY  'CH00009 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09414060 09414060 09414060
  VAPPL1         DEVLU2X         XMIT  B-ONLY  'CH00009 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09414060 09414060 09414060
  VAPPL2         DEVLU2          RECV  E-ONLY  5C00009 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09414070 09414070 09414070
  VAPPL1         DEVLU2X         RECV  E-ONLY  5C00009 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09414070 09414070 09414070
  VAPPL1         DEVLU2X         XMIT  B-ONLY  'CH00010 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09414400 09414400 09414400
  VAPPL2         DEVLU2          XMIT  B-ONLY  'CH00010 >>> SECONDARY MESSAGEABCDEFGHIJKLMNOPQRST  00     09414400 09414400 09414390
  VAPPL2         DEVLU2          RECV  E-ONLY  5C00010 <<< PRIMARY MESSAGEABCDEFGHIJKLMNOPQRSTUVW  00     09414400 09414400 09414400
```

Figure 30. The Response Time Utility Listing of Transaction Records

```
                                                  RESPONSE TIME FREQUENCY DISTRIBUTION

NETWORK  SAMPNET      80 |-------------------------------------------------------------------------------------------------------|
VTAMAPPL VAPPL1          |                                                                                                       |
TERMINAL DEVLU2X         |                                                                                                       |
TRANSACT ONLY            |                                                                                                       |
                         |                                                                                                       |
                      70 |-------------------------------------------------------------------------------------------------------|
                         |                                                                                                       |
                         |                                                                                                       |
                         |                                                                                                       |
                         |                                                                                                       |
                      60 |-------------------------------------------------------------------------------------------------------|
                         |                                                                                                       |
                         |                                                                                                       |
                         |                                                                                                       |
                         |                                                                                                       |
         PERCENTAGE   50 |-------------------------------------------------------------------------------------------------------|
                         |                                                                                                       |
             OF          |                                                                                                       |
                         |                                                                                                       |
         RESPONSES       |                                                                                                       |
                      40 |-------------------------------------------------------------------------------------------------------|
                         |                                                                                                       |
                         |                                                                                                       |
                         |                                                                                                       |
                         |                                                                                                       |
                      30 |-------------------------------------------------------------------------------------------------------|
                         |                                                                                                       |
                         |                                                                                                       |
                         |                                                                                                       |
                         |                                                                                                       |
                      20 |-------------------------------------------------------------------------------------------------------|
                         |     *                                                                                                 |
                         |     *                                                                                                 |
                         |     *                                                                                                 |
                         |     *                                                                                                 |
                      10 |-----*-------------------------------------------------------------------------------------------------|
                         |   * * *                                                                                               |
                         |   * * *       *                                                                                       |
                         |   * * * *   *   *               *     *                                                               |
                         |   * * * * * * * * *     * * * * * * * * * * * * * * * * *                                             |
                        _________________________________________________________________________________________________________
                             | + + + + | + + + + | + + + + | + + + + | + + + + | + + + + | + + + + | + + + + | + + + + | + + + +
                            0.0       1.0       2.0       3.0       4.0       5.0       6.0       7.0       8.0       9.0
                                                            RESPONSE TIME (SECONDS)         INCREMENT = 0.2 SECONDS
                                                                                            PERCENTAGE ABOVE LAST INCREMENT = 0.3
```

Figure 31. The Response Time Utility Response Time Frequency Distribution

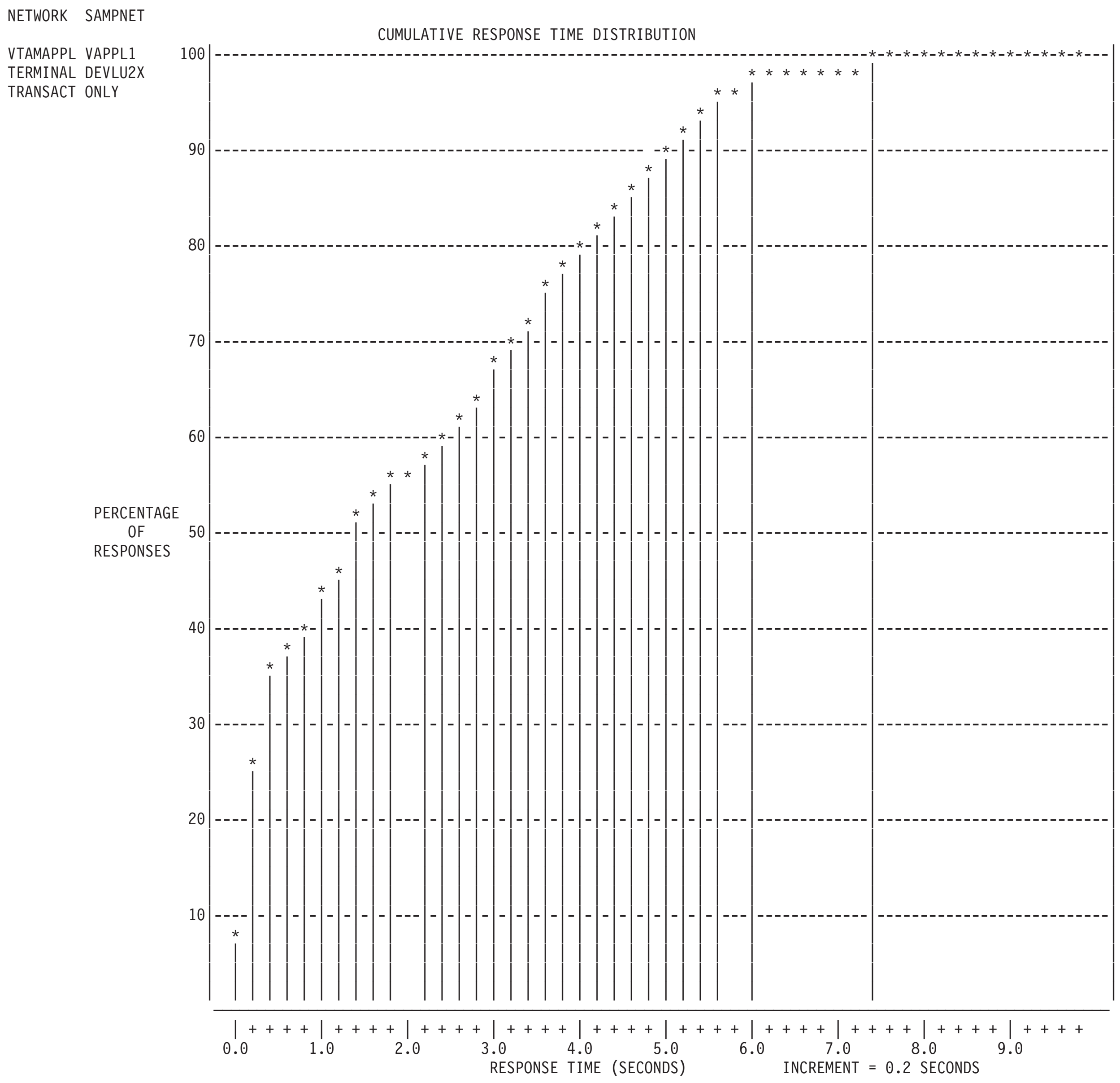


Figure 32. The Response Time Utility Cumulative Response Time Distribution

```
NETWORK  SAMPNET                                     TIME GRAPH OF RESPONSES
VTAMAPPL VAPPL1                                                                                                      < MINIMUM
TERMINAL DEVLU2X                                                                             INTERVAL =    30 SEC     * AVERAGE
TRANSACT ONLY                                                                                INCREMENT=   0.1 SEC     > MAXIMUM
                          RESPONSE TIME (SECONDS)
                        |                                                                                                     |
   TIME     NUMBER OF   |                                                                                                    1|
            RESPONSES   |0----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0|
  9.40.30        0      |                                                                                                     |
  9.41.00        0      |                                                                                                     |
  9.41.30        5      |<------------*------------------------------------------>                                            |
  9.42.00        8      | <------------*-------------->                                                                       |
  9.42.30        3      |  <---------------------------------------------------------------------*---------------------------->
  9.43.00        8      | <----------*-------------------------->                                                             |
  9.43.30        6      |<---------------*---------------------->                                                             |
  9.44.00        4      |<--------------------------*----------------------->                                                 |
  9.44.30        6      | <----------------------------------*------------------------>                                       |
  9.45.00        6      |<------------------*----------------------->                                                         |
  9.45.30        6      | <-------------------------*-------------------------->                                              |
  9.46.00        7      | <------------*----------------------------------->                                                  |
  9.46.30        6      |<--------*------->                                                                                   |
  9.47.00        5      |<----------*------------------->                                                                     |
  9.47.30        6      |<----------------------------------*------------------------------------------------->               |
  9.48.00        6      |       <----------------*--------------------------->                                                |
  9.48.30        7      |<-----------------------*------------------------------------->                                      |
  9.49.00        5      |       <-------------------*---------------------------------->                                      |
  9.49.30        8      |<---------------------*---------------------------------------->                                     |
  9.50.00        8      |<-------*-------------------->                                                                       |
  9.50.30        8      |<-----------------*----------------------->                                                          |
  9.51.00        8      |<----------------*--------------------------------->                                                 |
  9.51.30        4      |<-*-->                                                                                               |
  9.52.00        0      |                                                                                                     |
  9.52.30        7      | <------------*------------------------->                                                            |
  9.53.00        7      | <-----------*--------------------->                                                                 |
  9.53.30        6      |         <----------------------------*--------------------------------->                            |
  9.54.00        7      | <-----------------*------------------------------>                                                  |
  9.54.30        8      | <-----------------*---------------------------------->                                              |
  9.55.00        5      |  <---------------------------*------------------------->                                            |
  9.55.30        6      |<------------------------------------*------------->                                                 |
  9.56.00        8      | <------------------*--------------------------------------->                                        |
  9.56.30        6      |  <--------------------------*------------------------->                                             |
  9.57.00        5      |                       <---------------*------------------>                                          |
  9.57.30        7      | <--------------------*-------------------------->                                                   |
  9.58.00        6      | <--------------------------*------------------------->                                              |
  9.58.30        9      | <-----------*-------------------->                                                                  |
  9.59.00        9      |<-----------*---------------->                                                                       |
  9.59.30        6      |    <-------------------*------------------------------>                                             |
 10.00.00        5      |    <----------------------------------*---------------------->                                      |
 10.00.30        7      |<----------------*----------------------->                                                           |
 10.01.00        5      |  <--------------------------------*--------------------------------------------------------->       |
 10.01.30        6      |<------------------------------*------------------------------->                                     |
 10.02.00        7      |<---------------*------------------------------->                                                    |
 10.02.30        7      |   <------------*----------------------------->                                                      |
 10.03.00        5      | <------------------*-------------------------------->                                               |
 10.03.30        6      |   <------------------*------------------------>                                                     |
 10.04.00        7      |<-----------------------*----------------------------->                                              |
                        | ----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+---- |
```

Figure 33. The Response Time Utility Time Graph of Responses

Understanding Response Time Utility return codes

The only return code the Response Time Utility sets is 0, meaning response times are logged as specified.

Chapter 9. Specifying Response Time Utility control commands

The Response Time Utility control commands determine what time limits to use for the analysis, which lines and terminals to evaluate, how to compute response times, and how to generate the output reports.

The RUN command indicates the end of a set of input commands. It causes the Response Time Utility to process the log data set records according to your specifications. You can specify multiple sets of input commands, each set ending with a RUN command, which causes the Response Time Utility to process the log data set more than once. The Response Time Utility lists each group of input commands ending with a RUN command in the Response Time Utility output before generating the reports according to those commands.

Some commands, if specified more than once before a RUN command, replace their previous values, and the run proceeds according to the latest values for the commands. In addition, the default values for these commands are used in a run, even if you do not specify the commands. The following commands replace their previous values:

- CGRAPH
- HEADER
- PERCENT
- TGRAPH
- TRUNC
- EXIT
- LENGTH
- PROCESS
- TIME
- UNLOCK
- GRAPH
- NTWRK
- REPORT
- TPRINT

The effects of the following commands, if specified more than once before a RUN command, are cumulative (values specified later do not override previous values):

- BTRANS
- ETRANS
- MSGTXT
- VTAMAPPL
- APPCLU
- TCPIP
- TERM
- EXTERM

The following sections provide an explanation of the control commands and operands you can use to operate the Response Time Utility. To help you use this information, they also explain the requirements to enter the coding and the coding conventions for control commands.

Coding the control commands

You can enter a command in any position of the input record. Operands cannot extend past column 71, and they cannot be continued. Although you can separate a command and its operand by more than one blank space, you must enter at least one blank space between the command and its operand.

You can abbreviate some of the Response Time Utility control commands to a single letter. For example,

```
T APPL1
```

is equivalent to

```
TERM APPL1
```

Note that the allowable abbreviations for the commands are listed underneath the full command keyword. You can code either the full keyword or the abbreviation.

You must code all control commands in uppercase when using the control command input file.

Understanding control command conventions

The control commands for the Response Time Utility use the same coding conventions as the Loglist Utility commands. Refer to "Understanding control command coding conventions" on page 45 for information about the coding conventions.

APPCLU—Define an APPC LU for response time analysis

```
APPCLU name
       [,REPORT=(option1[,option2[,...]])]
A name
  [,REPORT=(option1[,option2[,...]])]
```

The APPCLU command specifies the name of an APPC LU for which a response time analysis will be done. It also defines the output options to be used when printing reports for the APPCLU.

If you code any VTAMAPPL, APPCLU or TCPIP commands, only those VTAM applications, APPC LUs and TCP/IP connections, will be included in the response time analysis. If you do not code any VTAMAPPL, APPCLU or TCPIP commands, all VTAM applications, APPC LUs and TCP/IP connections encountered on the log data set during the defined time interval will be processed.

name

Function: The *name* operand specifies the APPC LU name for which response time analysis is to be done.

Format: The value of the *name* operand can be any 1- to 8-character APPCLU name.

Default: None. This operand is required.

REPORT=(*option1*[,*option2*[,...]])

Function: The REPORT operand specifies the output report options for this APPC LU.

Note: Any options specified in this operand override the corresponding options specified in the TERMGRP operand of the REPORT command. This operand is ignored if LEVEL=SUMMARY is specified on the REPORT command.

Format: For the REPORT operand, you can code one or more option keywords within the parentheses in any order. For more information on the valid option keywords, refer to the TERM operand of the REPORT command.

Default: This operand is optional. If it is omitted, the output options for this APPCLU will be taken from the TERMGRP operand of the REPORT command.

BTRANS—Begin transaction definition

```
BTRANS TYPE=name
       [,LENG={integer|1}]
       [,LOC={TH+n|RH+n|RU+n|integer|1}]
       [,LOG=byte]
       [,MATCH={FIRST|LAST}]
       [,RECORD={XMIT|RECV}]
       [,SCAN={YES|integer|1}]
       [,SNA={YES|NO}]
       [,TEXT={(data)|'xx'}]
       [,TIME={READY|START|STOP}]
```

The BTRANS command defines the characteristics of a transmit or receive record that will make the record the beginning of a logical transaction for a terminal.

TYPE=*name*

Function: The TYPE operand specifies the name that is used to identify the transaction type being defined.

Format: The value of *name* can be from 1 to 8 nonblank characters.

Default: None. This operand is required.

LENG={*integer*|1}

Function: The LENG operand specifies the minimum number of data characters that a log record must have to be considered for beginning a transaction.

Format: The value of *integer* can be any integer from 1 to 32000.

Default: 1.

Note: The length of log records for SNA devices is taken to be only the RU portions of the records.

LOC={TH+*n*|RH+*n*|RU+*n*|*integer*|1}

Function: The LOC operand specifies the starting byte in the data portion of the log record where the comparison is to take place.

Note: This operand is valid only if the TEXT operand is also coded.

Format: You can code one of the following values for the LOC operand:

TH+*n* Indicates an offset from the start of the transaction header (TH) where *n* is an integer from 0 to 32000. TH+0 is the first byte of the TH.

RH+*n* Indicates an offset from the start of the response header (RH) where *n* is an integer from 0 to 32000. RH+0 is the first byte of the RH.

RU+*n* Indicates an offset from the start of the request/response unit (RU) where *n* is an integer from 0 to 32000. RU+0 is the first byte of the RU.

integer Indicates the relative byte in the RU (or data stream for non-SNA) that will be the starting position of the test. If *integer* is greater than the length of the data portion of the record, the record will not be considered.

Default: 1.

Coding the TH+*n*, RH+*n*, or RU+*n* values for non-SNA messages causes the LOC value to be the same as LOC=*integer* where *integer* = *n*+1. For example:

- LOC=RU+0 is equivalent to LOC=1 for non-SNA messages
- LOC=TH+3 is equivalent to LOC=4 for non-SNA messages.

LOG=*byte*

Function: The LOG operand specifies the byte to be compared with the contents of the user data field in the header of a log record. The log byte can be set using the TRANSMIT, INITSELF, and SNACMND statements in STL, and using the TEXT and CMND statements in the WSim Scripting Language.

Note: Either the LOG or the TEXT operand must be coded. If both LOG and TEXT are specified, the LOG operand has priority. The comparison for the user data byte will be performed first, and if there is a match, the record is accepted regardless of the data specified by the TEXT operand.

Format: You can enter one byte of data. The Response Time Utility treats a single digit as an EBCDIC character and two digits as a hexadecimal character.

MATCH={FIRST|LAST}

Function: The MATCH operand specifies whether the first or last of a series of consecutive records that match the BTRANS specifications is to be used to begin a response time for a terminal.

Format: You can code one of the following values for the MATCH operand:

FIRST The first record that matches the BTRANS specifications will be used to begin a response time.

LAST The last record that matches the BTRANS specifications will be used to begin a response time.

Default: FIRST.

RECORD={XMIT|RECV}

Function: The RECORD operand specifies whether the record beginning this transaction is to be a XMIT record (transmitted by WSim) or a RECV record (received by WSim).

Format: For the RECORD operand, you can code XMIT or RECV.

Default: XMIT.

SCAN={YES|*integer*|1}

Function: The SCAN operand specifies that a record is to be scanned sequentially for the data specified in the TEXT operand. The data starting at the location specified by the LOC operand is scanned byte by byte and compared with the character string specified by the TEXT operand. If data is found that matches the TEXT data before the specified number of positions have been scanned, the record is accepted.

Note: This operand is valid only if the TEXT operand is also coded.

Format: You can code the following values for the SCAN operand:

YES Scanning continues until the TEXT data is matched or the end of the record is reached.

integer An integer from 1 to 32000. It specifies that scanning continues until the TEXT data is matched, the specified number of positions have been scanned, or the end of the record is reached.

Default: 1.

SNA={YES|NO}

Function: The SNA operand specifies whether all SNA transmit and receive records are to be considered for transaction processing.

Note: This operand is ignored for non-SNA data.

Format: You can code one of the following values for the SNA operand:

YES All SNA records are considered for transaction processing, regardless of SNA message content.

NO Only FM data that is not a SNA response is considered for transaction processing.

Default: NO.

TEXT={(*data*)|'*xx*'}

Function: The TEXT operand defines the data to be used as the comparison data or the mask to be used as the comparison mask when scanning a log record.

Note: Either the TEXT or the LOG operand must be coded. If both LOG and TEXT are coded, the LOG operand has priority.

Format: You can code one of the following values for the TEXT operand:

(*data*) Is 1 to 50 bytes of data enclosed in parentheses. You can enter hexadecimal data within the parentheses by coding the digits within single quotes. To use a parenthesis or single quote as a data byte, code two of the characters. If you code (*data*), messages will be searched for contents that exactly match the data provided.

Note: SO and SI characters are removed from DBCS data entered. To include an SO or SI character in the data for the data compare, add the hexadecimal equivalent of an SO (X'0E') or SI (X'0F') to the data.

'*xx*' Is two hexadecimal digits contained within single quotes. If '*xx*' is specified, a test under mask, which compares the byte indicated by the LOC operand with the 2 hexadecimal digits within single quotes, will be made. Since only 1 byte is compared, the SCAN operand does not apply and is ignored. If all bits that are 1 in the mask byte ('*xx*') are also 1 in the data byte tested, then a match is found.

Default: None. This operand is optional.

TIME={READY|START|STOP}

Function: The TIME operand specifies which time stamp is to be used in calculating response times involving records selected by this command.

Format: For the TIME operand, you can code READY, START, or STOP.

Default: READY.

CGRAPH—Define scale for cumulative distribution graph

```
CGRAPH [INCR={integer|0}]
       [,LINES={YES|NO}]
```

The CGRAPH command defines the scale factor for the cumulative distribution graphs that can be printed after the terminal, terminal group, and summary output reports.

Specifying the CGRAPH command will not cause a cumulative distribution graph to be printed. To print the graph, use the REPORT command or the REPORT operand on the VTAMAPPL, APPCLU, TCPIP, and TERM commands.

INCR={*integer*|0}

Function: The INCR operand specifies the increment value to be used between response times (in tenths of seconds).

Note: If a value of 0 is coded or defaulted, the increment value will be computed by subtracting the smallest response time from the largest response time and dividing the difference by 50.

Format: The value of *integer* can be any integer from 0 to 999.

Default: 0.

LINES={YES|NO}

Function: The LINES operand determines whether the highlighting lines, indicating jumps in the curve, will be printed.

Format: For the LINES operand, you can code YES or NO.

Default: NO.

END—End response time processing

```
END
```

The END command specifies that the Response Time Utility is to terminate normally. No further processing is done, and all open data sets are closed. Any commands entered after the last RUN command, but before the END command, are ignored.

ETRANS—End transaction definition

```
ETRANS {LOG=byte}
       {,TEXT={(data)|'xx'}}
       [,LENG={integer|1}]
       [,LOC={TH+n|RH+n|RU+n|integer|1}]
       [,MATCH={FIRST|LAST}]
       [,RECORD={XMIT|RECV}]
       [,SCAN={YES|integer|1}]
       [,SNA={YES|NO}]
       [,TIME={READY|START|STOP}]
```

The ETRANS command defines the characteristics of a transmit or receive record that will make it the end of a logical transaction for a terminal.

LOG=*byte*

Function: The LOG operand specifies the byte to be compared with the contents of the user data field in the header of a log record. The log byte can be set using the TRANSMIT, INITSELF, and SNACMND statements in STL, and using the TEXT and CMND statements in the WSim Scripting Language.

Note: Either the LOG or the TEXT operand must be coded. If both LOG and TEXT are coded, the LOG operand has priority. The comparison for the user data byte will be performed first, and if there is a match, the record is accepted regardless of the data specified by the TEXT operand.

Format: You can enter one byte of data. The Response Time Utility treats a single digit as an EBCDIC character and two digits as a hexadecimal character.

TEXT={(*data*)|'*xx*'}

Function: The TEXT operand defines the data to be used as the comparison data or the mask to be used as the comparison mask when scanning a log record.

Note: Either the TEXT or the LOG operand must be coded. If both are coded, the LOG operand has priority.

Format: You can code one of the following values for the TEXT operand:

(*data*) Is 1 to 50 bytes of data enclosed in parentheses. You can enter hexadecimal data within the parentheses by coding the digits within single quotes. To use a parenthesis or single quote as a data byte, code two of the characters. If you code (*data*), messages will be searched for contents which exactly match the data provided.

Note: SO and SI characters are removed from DBCS data entered. To include an SO or SI character in the data for the data compare, add the hexadecimal equivalent of an SO (X'0E') or SI (X'0F') to the data.

'*xx*' Is two hexadecimal digits contained within single quotes. If '*xx*' is specified, a test under mask, which compares the byte indicated by the LOC operand with the 2 hexadecimal digits within single quotes, will be made. Since only 1 byte is compared, the SCAN operand does not apply and is ignored. If all bits that are 1 in the mask byte ('*xx*') are also 1 in the data byte tested, then a match is found.

Default: None. This operand is optional.

LENG={*integer*|1}
Function: The LENG operand specifies the minimum number of data characters that a log record must have to be considered for ending a transaction.

Note: The length of log records for SNA devices is taken to be only the RU portions of the records.

Format: The value of *integer* can be any integer from 1 to 32000.

Default: 1.

LOC={TH+*n*|RH+*n*|RU+*n*|*integer*|1}
Function: The LOC operand specifies the starting byte in the data portion of the log record where the comparison is to take place.

Note: This operand is valid only if the TEXT operand is also coded.

Format: You can code one of the following values for the LOC operand:

TH+*n* Indicates an offset from the start of the transaction header (TH) where *n* is an integer from 0 to 32000. TH+0 is the first byte of the TH.

RH+*n* Indicates an offset from the start of the response header (RH) where *n* is an integer from 0 to 32000. RH+0 is the first byte of the RH.

RU+*n* Indicates an offset from the start of the request/response unit (RU) where *n* is an integer from 0 to 32000. RU+0 is the first byte of the RU.

integer Indicates the relative byte in the RU (or data stream for non-SNA) that will be the starting position of the test. If *integer* is greater than the length of the data portion of the record, the record will not be considered.

Default: 1.

Coding the TH+*n*, RH+*n*, or RU+*n* values for non-SNA messages causes the LOC value to be the same as LOC=*integer* where *integer* =*n*+1. For example:
- LOC=RU+0 is equivalent to LOC=1 for non-SNA messages
- LOC=TH+3 is equivalent to LOC=4 for non-SNA messages.

MATCH={FIRST|LAST}
Function: The MATCH operand specifies whether the first or last of a series of consecutive records that match the ETRANS specifications is to be used to end the response time for a terminal.

Note: ETRANS MATCH=LAST may not be coded if its corresponding BTRANS entry also has MATCH=LAST. Otherwise, unpredictable results can occur in this situation.

Format: You can code one of the following values for the MATCH operand:

FIRST The first record that matches the ETRANS specifications will be used to end a response time.

LAST The last record that matches the ETRANS specifications will be used to end a response time.

Default: FIRST.

RECORD={XMIT|RECV}

Function: The RECORD operand specifies whether the record ending this transaction is to be a XMIT record (transmitted by WSim) or a RECV record (received by WSim).

Format: For the RECORD operand, you can code XMIT or RECV.

Default: RECV.

SCAN={YES|*integer*|1}

Function: The SCAN operand specifies that a record is to be scanned sequentially for the data specified in the TEXT operand. The data starting at the location specified by the LOC operand is scanned byte by byte and is compared with the character string specified by the TEXT operand. If data is found that matches the TEXT data before the specified number of positions have been scanned, the record is accepted.

Note: This operand is valid only if the TEXT operand is also coded.

Format: You can code the following values for the SCAN operand:

YES Scanning continues until the TEXT data is matched or the end of the record is reached.

integer An integer from 1 to 32000. It specifies that scanning continues until the TEXT data is matched, the specified number of positions have been scanned, or the end of the record is reached.

Default: 1.

SNA={YES|NO}

Function: The SNA operand specifies whether all SNA transmit and receive records are to be considered for transaction processing.

Note: This operand is ignored for non-SNA data.

Format: You can code one of the following values for the SNA operand:

YES All SNA records are considered for transaction processing, regardless of SNA message content.

NO Only FM data that is not a SNA response is considered for transaction processing.

Default: NO.

TIME={READY|START|STOP}

Function: The TIME operand specifies which time stamp is to be used in calculating response times involving records selected by this command.

Format: For the TIME operand, you can code READY, START, or STOP.

Default: READY.

EXIT—Define user exit

```
EXIT member
     [,PARM=(data)]
```

The EXIT command specifies the name of a user exit routine that will calculate response times. For more information about calculating response times, refer to

"Calculating response times for terminals" on page 113. See *WSim User Exits* for more information about user exit routines.

member

Function: The *member* operand is the name of the user exit routine to be loaded by the Response Time Utility and given control each time a log record is read that satisfies the other command specifications.

Note: If you do not code the EXIT command, a WSim-supplied exit is invoked to calculate the response times. If you code the EXIT command, but omit the *member* operand, any currently loaded user exit is deleted, and the WSim-supplied exit is invoked.

Format: The value of *member* can be any 1- to 8-character name that conforms to standard JCL naming conventions.

Default: None. This operand is optional.

PARM=(*data***)**

Function: The PARM operand specifies the user parameter data to be passed to the user exit routine each time it is called to process a log record.

Note: If this operand is coded, it must be preceded by the *member* operand.

Format: The *data* for the PARM operand can be from 1 to 50 EBCDIC characters. You do not have to duplicate special characters.

Default: None. This operand is optional.

EXTERM—Exclude terminal

```
EXTERM name[-num]
EXT name[-num]
```

The EXTERM command specifies the name of the device, logical unit (LU), or transaction program (TP) for which the analysis will not be performed.

You must enter an EXTERM command for each individual device, LU, or TP which you would not like in the Response Time Utility listing. If the EXTERM command does not follow a valid TCPIP, VTAMAPPL, or APPCLU command, the *name* specified will not be listed for all TCP/IP connections, VTAM applications, and APPC LUs.

If the EXTERM command follows a valid TCPIP, VTAMAPPL, or APPCLU command, the *name* specified will not be listed for that TCP/IP connection, VTAM application, or APPC LU only.

name

Function: The *name* operand specifies the name of the device, logical unit (LU), or transaction program (TP).

Format: The value of the *name* operand is a 1- to 8-character name that matches the name coded on a WSim network definition DEV, LU, or TP statement.

num

Function: The *num* operand specifies either a single session number for an LU with multiple session capability or a specific transaction program instance.

Format: For an LU, the *num* operand can be any decimal integer from 1 to 65535. For TP instances, the *num* operand can be any decimal integer from 1 to 99999.

Note: If *num* is not appended to the LU name and multiple sessions exist for the LU, all of the sessions are excluded. If *num* is not appended to the TP name and multiple instances of the TP exist, all instances are excluded.

GRAPH—Define scale for frequency distribution graph

```
GRAPH [INCR={n|0}]
```

The GRAPH command defines the scale factor for the frequency distribution graphs that can be printed after the terminal, terminal group, and summary output reports.

Specifying the GRAPH command will not cause a frequency distribution graph to be printed. To print the graph, use the REPORT command or the REPORT operand on the VTAMAPPL, APPCLU, TCPIP, and TERM commands.

INCR={*n*|0}

Function: The INCR operand specifies the increment value to be used between response times (in tenths of seconds).

Note: If a value of 0 is coded or defaulted, the increment value will be computed by subtracting the smallest response time from the largest response time and dividing the difference by 50.

Format: The value of *n* can be any integer from 0 to 999.

Default: 0.

HEADER—Define output report header

```
HEADER data
```

The HEADER command defines the data to be used as the page header on the Response Time Utility output reports.

data

Function: The *data* operand specifies the data to be used as the page header on the Response Time Utility output reports.

Format: The *data* operand can be up to 27 characters, including blanks and special characters. The *data* operand begins with the first nonblank character after the HEADER command and ends with the characters in column 71 or after 27 characters.

Default: The default value is “WSim RESPONSE TIME ANALYSIS”.

LENGTH—Define minimum record lengths

```
LENGTH [RECEIVE={integer|1}]
       [,TRANSMIT={integer|1}]
```

The LENGTH command specifies the minimum length of transmit and receive records that can be used in computing response times. This command is used in conjunction with normal response time rules (PROCESS SYSTEM or PROCESS ACTUAL).

The LENGTH command does not affect user-defined transactions specified by BTRANS and ETRANS commands. However, when a transaction is defined by a BTRANS command only (for example, the ETRANS command is omitted, and the response time ends according to the normal rules), the RECEIVE operand value is used.

RECEIVE={*integer*|1}
Function: The RECEIVE operand specifies the minimum number of data characters that a receive log record must have to be considered for ending a response time.

Note: The length of log records for SNA devices is taken to be only the RU portions of the records.

Format: The value of *integer* can be any integer from 1 to 32000.

Default: 1.

TRANSMIT={*integer*|1}
Function: The TRANSMIT operand specifies the minimum number of data characters that a transmit log record must have to be considered for beginning a response time.

Note: The length of log records for SNA devices is taken to be only the RU portions of the records.

Format: The value of *integer* can be any integer from 1 to 32000.

Default: 1.

MSGTXT—Define a message generation deck for response time analysis

```
MSGTXT nameM name
```

The MSGTXT command specifies the name of a message generation deck or STL procedure for which further selection of records will be done. If particular resources are requested for the run, only those records that match the specified message generation decks or STL procedures will be used. If more than one MSGTXT command is entered, all specified message generation deck or STL procedure names will be used in the selection of records.

name

Function: The *name* operand specifies a name of a message generation deck or STL procedure for which selection of records will be done.

Format: The value of the *name* operand is a 1- to 8-character name that matches the name field from a WSim message generation or STL MSGTXT statement.

Note: When a transaction begins in one message generation deck or STL procedure and ends in another, unexpectedly long response time calculations can result if only the first message generation deck or STL procedure has been specified on a MSGTXT command. To achieve realistic calculations, you must specify both message generation decks or STL procedures.

NTWRK—Define a network for response time analysis

```
NTWRK name
N name
```

The NTWRK command specifies the network name for which a response time analysis will be done.

If you do not code a NTWRK command, the response time analysis will be done for all networks that have records on the log data set.

name

Function: The *name* operand specifies the name of the network for which a response time analysis will be done.

Format: The value of the *name* operand can be any 1- to 8-character name that must match the label name specified on a WSim NTWRK network definition statement.

Default: None. This operand is required.

P—Terminate console input

```
P
```

The P command specifies that command input from the operator's console is to be terminated and that the Response Time Utility is to begin reading commands from the SYSIN data set. If you enter this command and the SYSIN data set is not open, the Response Time Utility continues requesting input from the console. The P command is ignored if encountered in the SYSIN data set.

PERCENT—Define percentile values

```
PERCENT {p1[,p2[,...]]|90}
```

The PERCENT command specifies the percentiles for which response times will be found.

{*p1*[,*p2*[,...]]|90}

Function: The operands specify the percentile values for which response times will be found. For each value specified, a response time will be found such that the specified percentage of all response times will be less than or equal to the found response time.

Format: The operand value can be any integer from 1 to 99. You can code up to 10 values separated by commas.

Default: 90.

PROCESS—Define response time type

```
PROCESS {ACTUAL|SYSTEM}
```

The PROCESS command specifies the type of response times to be computed when user-defined transactions are not specified. Details on how response times are computed are in “Calculating response times for terminals” on page 113.

{ACTUAL|SYSTEM}

Function: The operand specifies the type of response times to be computed.

Format: The operand value can be either ACTUAL or SYSTEM.

Default: SYSTEM.

Note: If PROCESS ACTUAL is specified for a large log data set and TRUNC YES is not specified, it may take a great deal of host processor time to run the Response Time Utility.

REPORT—Define output options

```
REPORT [LEVEL={TERM|TERMGRP|SUMMARY}]
       [,TERM=(option1[,option2[,...]])]
       [,TERMGRP=(option1[,option2[,...]])]
       [,SUMMARY=(option1[,option2[,...]])]
```

The REPORT command defines the types of output reports that will be printed for a response time analysis.

LEVEL={TERM|TERMGRP|SUMMARY}

Function: The LEVEL operand specifies the level of output reports to be printed.

Format: You can code one of the following values for the LEVEL operand:

TERM	A response time report will be printed for each terminal, each terminal group (TCP/IP connection, VTAM application, or APPC LU) and for the summary of all terminals.
TERMGRP	A response time report will be printed for each terminal group and for the summary of all terminals.
SUMMARY	A report summarizing the response times for all terminals will be printed.

Default: TERM.

TERM=(*option1***[,***option2***[,...]])**

Function: The TERM operand specifies the default output options for terminal reports.

Note: These output options can be overridden for a specific terminal by coding the REPORT operand for a TERM command. This operand is ignored if LEVEL=TERMGRP or LEVEL=SUMMARY is coded.

Format: For the TERM operand, you can code one or more option keywords within the parentheses in any order. The valid option keywords are:

CGRAPH	Print a cumulative distribution graph.
NOCGRAPH	Do not print a cumulative distribution graph.
GRAPH	Print a frequency distribution graph.
NOGRAPH	Do not print a frequency distribution graph.
LIST	Print a list of the computed response times.
NOLIST	Do not print a list of the computed response times.
TGRAPH	Print a time graph.
NOTGRAPH	Do not print a time graph.
TRANS	Print a report for each defined transaction type.
NOTRANS	Print a single report for all transaction types.

Default: LIST, NOGRAPH, NOCGRAPH, NOTGRAPH, and TRANS.

TERMGRP=(*option1***[,***option2***[,...]])**

Function: The TERMGRP operand specifies the default output options for terminal group reports.

Note: These output options can be overridden for a specific terminal group by coding the REPORT operand for a TCPIP, VTAMAPPL, or APPCLU command. This operand is ignored if LEVEL=SUMMARY is coded.

Format: For the TERMGRP operand, you can code one or more option keywords within the parentheses in any order. For more information on the valid option keywords, refer to the TERM operand.

Default: LIST, NOGRAPH, NOCGRAPH, NOTGRAPH, and TRANS.

SUMMARY=(*option1***[,***option2***[,...]])**

Function: The SUMMARY operand specifies the output options for the summary report.

Format: For the SUMMARY operand, you can code one or more option keywords within the parentheses in any order. For more information on the valid option keywords, refer to the TERM operand.

Default: LIST, NOGRAPH, NOCGRAPH, NOTGRAPH, and TRANS.

RUN—Perform response time analysis

```
RUN
```

The RUN command specifies that all commands have been entered and that processing of the log data set should begin. You must enter this command to start the response time analysis. After processing is complete, all parameters are reset to their default values before any further commands are interpreted. If RUN is entered and no other commands have been specified, response times will be computed for all terminals on the log data set.

If consecutive runs specify time limits that are in ascending order and that do not overlap, the log data set will not be closed and reopened between the runs. If a run specifies TIME ALL, TIME START-END, or a time interval that includes midnight, the data set will be closed and reopened before the next run.

TCPIP—Define a TCP/IP connection for response time analysis

```
TCPIP name
     [,REPORT=(option1[,option2[,...]])]
```

The TCPIP command specifies the name of a TCP/IP connection for which a response time analysis will be done. It also defines the output options to be used when printing reports for the TCP/IP connection.

If you code any TCPIP, APPCLU, or VTAMAPPL commands, only those TCP/IP connections, APPC LUs, and VTAM applications will be included in the response time analysis. If you do not code any TCPIP, APPCLU, or VTAMAPPL commands, all TCP/IP connections, APPC LUs, and VTAM applications encountered on the log data set during the defined time interval will be processed.

name

Function: The *name* operand specifies the TCP/IP connection name for which response time analysis is to be done.

Format: The value of the *name* operand can be any 1- to 8-character TCPIP name.

Default: None. This operand is required.

REPORT=(*option1***[,***option2***[,...]])**

Function: The REPORT operand specifies the output report options for this TCP/IP connection.

Note: Any options specified in this operand override the corresponding options specified in the TERMGRP operand of the REPORT command. This operand is ignored if LEVEL=SUMMARY is specified on the REPORT command.

Format: For the REPORT operand, you can code one or more option keywords within the parentheses in any order. For more information on the valid option keywords, refer to the TERM operand of the REPORT command.

Default: This operand is optional. If it is omitted, the output options for this TCP/IP connection will be taken from the TERMGRP operand of the REPORT command.

TERM—Define a terminal for response time analysis

```
TERM name[-num]
     [,REPORT=(option1[,option2[,...]])]
T name[-num]
  [,REPORT=(option1[,option2[,...]])]
```

The TERM command specifies the name of a terminal for which a response time analysis will be done. It also defines the output options to be used when printing reports for the terminal.

A TERM command must follow a valid TCPIP, APPCLU, or VTAMAPPL command; it applies to the last TCPIP, APPCLU, or VTAMAPPL command entered. CPI-C transaction programs are specified by APPCLU and TERM commands. VTAMAPPL LUs are specified by VTAMAPPL and TERM commands. TCP/IP client devices are specified by TCPIP and TERM commands. If you do not code any TERM commands after a TCPIP, APPCLU, or VTAMAPPL command, all terminals for the specified TCP/IP connection, APPC LU, or VTAM application will be included in the analysis.

name

Function: The *name* operand specifies the name of a simulated terminal for which the response time analysis is to be done.

Format: The value of the *name* operand can be any 1- to 8-character name that matches the name field from a WSim DEV, LU, or TP network definition statement.

Default: None. This operand is required.

num

Function: The *num* operand specifies a single session number for an LU with multiple session capability or a specific transaction program instance.

Format: For an LU, the value of *num* can be any integer from 1 to 65535. If *num* is not appended to the LU name and multiple sessions exist for the LU, response times will be computed for all of the sessions. For a TP, the value of *num* can be any integer from 1 to 99999. If *num* is not appended to the TP name and multiple instances exist for the TP, response times will be computed for all of the instances.

REPORT=(*option1***[,***option2***[,..]])**

Function: The REPORT operand specifies the output report options for this terminal.

Note: Any options specified in this operand override the corresponding options specified in the TERM operand of the REPORT command. This operand is ignored if LEVEL=TERMGRP or LEVEL=SUMMARY is specified on the REPORT command.

Format: For the REPORT operand, you can code one or more option keywords within the parentheses in any order. For more information on the valid option keywords, refer to the TERM operand of the REPORT command.

Default: This operand is optional. If it is omitted, the output options for this terminal will be taken from the TERM operand of the REPORT command.

TGRAPH—Define time graph parameters

```
TGRAPH [INCR={1|2|5|10}]
       [,INTERVAL={integer|10}]
       [,LINES={YES|NO}]
       [,ORIGIN={integer|0}]
       [,THRESH=integer]
```

The TGRAPH command defines the parameters for the time graphs that can be printed after the terminal, terminal group, and summary output reports.

Specifying the TGRAPH command will not cause a time graph to be printed. To print the graph, use the REPORT command or the REPORT operand on the VTAMAPPL, APPCLU, TCPIP, and TERM commands. No time graphs will be printed if you specify a user exit routine by an EXIT command.

INCR={1|2|5|10}

Function: The INCR operand specifies the increment value to be used in plotting the response times (in tenths of seconds).

Format: The value of the INCR operand can be 1, 2, 5, or 10.

Default: 1.

INTERVAL={*integer*|10}

Function: The INTERVAL operand specifies the time interval during which response values will be accumulated (in seconds).

Format: The value of the INTERVAL operand can be any integer from 1 to 3600.

Default: 10.

LINES={YES|NO}

Function: The LINES operand specifies whether the highlighting lines, showing the range of response times, will be printed.

Format: The value of the LINES operand can be YES or NO.

Default: NO.

ORIGIN={*integer*|0}

Function: The ORIGIN operand specifies the origin or lower limit of the response time scale of the time graph (in seconds).

Format: The value of the ORIGIN operand can be any integer from 0 to 500.

Default: 0.

THRESH=*integer*

Function: The THRESH operand specifies that a threshold line is to be printed on the time graph. The *integer* value specifies the increment value at which the threshold line is to be printed (in tenths of seconds).

Format: The *integer* value can be any integer from 1 to 32000.

Default: None. If this operand is omitted from the TGRAPH statement, no threshold line will be printed.

TIME—Specify time limits

```
TIME {ALL}
     {x-y}

     where x can be hhmmss, hhmm, or START
     and y can be hhmmss, hhmm, or END
```

The TIME command specifies the time limits of a simulation run for which the response time analysis will be performed. The first field indicates the time at which the analysis is to begin. This time is compared against the ready time of the log record. The second field indicates the time at which the analysis is to end. This time is compared against the stop time of the log record.

Times are 24-hour clock times (000000-235959). A time interval including midnight is valid, such as TIME 235000-004500. The log data set must contain records between a time that you specify and the beginning of the next hour. The first field indicates the time at which the analysis is to begin. The second field indicates the last second to be processed in the run. Except when using TIME ALL, any format of the first operand field can be paired with any format of the second operand field, for example, TIME 183000-END.

If you do not enter the TIME command, the entire log data set is processed. If you enter multiple TIME commands, the limits from the last TIME command entered are used.

ALL
: **Function:** The ALL operand specifies that the entire log data set is to be analyzed.

hhmm
: **Function:** The *hhmm* operand indicates the hour and minutes that limit the processing.

hhmmss
: **Function:** The *hhmmss* operand indicates the hours, minutes, and seconds that limit the processing.

START
: **Function:** The START operand indicates that analysis is to begin with the first record on the log data set.

END
: **Function:** The END operand indicates that analysis is to end with the last record on the log data set.

TPRINT—Print list of transaction records

```
TPRINT {YES|NO}
```

The TPRINT command determines whether or not a listing of log records considered for transaction processing will be printed. Up to 60 data characters from each log record will be printed. The records that begin and end valid transactions are identified by transaction type name.

{YES|NO}
Function: The operand value determines whether or not the listing of transaction records will be printed.

Format: For the operand value, you can code YES or NO.

Default: NO.

TRUNC—Truncate time stamps

```
TRUNC {YES|NO}
```

The TRUNC command determines whether or not the time stamps from the log records will be truncated to tenths of seconds before response time calculations are made.

{YES|NO}
Function: The operand value determines whether or not the time stamps will be truncated.

Note: If PROCESS ACTUAL is specified for a large log data set and TRUNC YES is not specified, it may take a great deal of host processor time to run the Response Time Utility.

Format: For the operand, you can code one of the following values:

YES The hundredths digit of the time stamp will be set to zero.

NO A time stamp will be accurate to a hundredth of a second.

Default: NO.

UNLOCK—Define use of keyboard unlock messages

```
UNLOCK {YES|NO}
```

The UNLOCK command determines whether or not keyboard unlock messages are to be considered as valid data messages when calculating response times for terminals using the 3270 data stream when not doing transaction processing.

{YES|NO}
Function: The operand value determines whether or not keyboard unlock messages will be used in calculating response times.

Format: For the operand, you can code the following values:

YES Keyboard unlock messages will be used in calculating response times.

NO Keyboard unlock messages will be discarded.

Default: NO.

VTAMAPPL—Define a VTAMAPPL for response time analysis

```
VTAMAPPL name
        [,REPORT=(option1[,option2[,...]])]
V name
  [,REPORT=(option1[,option2[,...]])]
```

The VTAMAPPL command specifies the name of a VTAMAPPL for which a response time analysis will be done. It also defines the output options to be used when printing reports for the VTAMAPPL.

If you code any TCPIP, APPCLU or VTAMAPPL commands, only those TCP/IP connections, APPC LUs, and VTAM applications will be included in the response time analysis. If you do not code any TCPIP, APPCLU or VTAMAPPL commands, allTCP/IP connections, APPC LUs, and VTAM applications encountered on the log data set during the defined time interval will be processed.

name

Function: The *name* operand specifies the VTAMAPPL name for which response time analysis is to be done.

Format: The value of the *name* operand can be any 1- to 8-character VTAMAPPL name.

Default: None. This operand is required.

REPORT=(*option1***[,***option2***[,...]])**

Function: The REPORT operand specifies the output report options for this VTAMAPPL.

Note: Any options specified in this operand override the corresponding options specified in the TERMGRP operand of the REPORT command. This operand is ignored if LEVEL=SUMMARY is specified on the REPORT command.

Format: For the REPORT operand, you can code one or more option keywords within the parentheses in any order. For more information on the valid option keywords, refer to the TERM operand of the REPORT command.

Default: This operand is optional. If it is omitted, the output options for this VTAMAPPL will be taken from the TERMGRP operand of the REPORT command.

*—Comment

```
* [comment]
```

The * command specifies a comment line that is listed with the other commands. It has no effect on the Response Time Utility processing.

comment

Function: Specifies a user comment that is printed with the Response Time Utility command listing. After the *, you can enter any data.

Chapter 10. Using ITPECHO to test WSim simulated resources

ITPECHO is a VTAM application program supplied with WSim as a sample routine. You can use ITPECHO with WSim simulated resources (the supplied sample message generation decks and network definition statements) to help you with the WSim installation, learning, and planning processes.

ITPECHO is essentially an "echo" program. It receives data and transmits it to the terminal that issued the request. ITPECHO supports 3270 terminals (LU2) and any non-3270 devices that do not have specific data stream dependencies (such as LU0).

The sections in this chapter present information about the following items:
- What requirements ITPECHO needs to run on your system
- How to install ITPECHO on your system
- How to run ITPECHO, including:
  - Execution parameters to use with JCL or a TSO CLIST
  - Examples of JCL and a TSO CLIST for running ITPECHO.
- Operator commands for starting ITPECHO
- Logon procedure for ITPECHO.

Understanding ITPECHO requirements

This section describes the functional requirements for ITPECHO, including:
- Programming requirements
- Storage requirements
- SNA considerations
- 3270 devices
- Non-3270 devices.

Programming requirements

ITPECHO runs with any current release of VTAM on currently supported MVS. It runs in either 24-bit or 31-bit addressing mode.

Storage requirements

The minimum virtual region size required to run ITPECHO is determined by the maximum number of concurrent sessions and the buffer size allocated for each session. To calculate the approximate storage requirements, use the following formula:

```
S = 14000 + N * ( 2*B + 480 )
```

where:

S	Storage requirements in bytes
N	Maximum number of expected concurrent sessions
B	BUFSIZE execution parameter value.

SNA considerations

Regardless of the real terminal type that is in session, the VTAM interface makes all terminals appear to ITPECHO as SNA logical units. Therefore, SNA protocols are used in the communication.

The BIND

A BIND image is used by both half-sessions at logon time to specify mutual operating procedures. Make sure you select an appropriate BIND image when you log on to ITPECHO. The following should be considered when you are selecting BIND parameters:

- Duplex, half-duplex flip-flop mode, and half-duplex contention are supported. If you specify other than half-duplex flip-flop mode for a 3270 LU2, ITPECHO changes the value to half-duplex flip-flop. Other terminal types will support all modes.
- ITPECHO obeys the primary response protocol setting (byte 4 bits 2-3). On every data request sent, that particular response type (none, exception, or definite) will be requested in the RH. If the response type is definite or exception (bits 2-3 = B'11'), the exception responses are always requested unless the PF6 function is requested (see below).
- ITPECHO never transmits multiple chain elements because of the static buffer allocations for each session. However, it can receive multiple chain elements from the terminal. When ITPECHO receives multiple chain elements, an “only-in-chain” element is echoed back to the LU, which contains the union of the individual elements received. The length of this element, like all others, is truncated, if necessary, to the size of the internal buffer (BUFSIZE).

  Note: If you are using VTAM Version 3, “only-in-chain” messages sent by ITPECHO to the terminal are automatically chained based on the primary maximum RU size in the BIND image for the session.

When an LU requests a session with ITPECHO, the LU should use a 3270 data stream. If the terminal is not a 3270, the INITIATE request should contain NON3270 as the user data field. When this is seen by ITPECHO, RUs are echoed exactly, with no command fields inserted.

Functions available with 3270 devices

Note: This section applies only to ITPECHO sessions with 3270 devices. ITPECHO provides more than just an echo capability for 3270 devices. A terminal can request a function by using a PF key, data stream content, or a combination of a PF key and data stream content. The following functions are available depending on the attention identifier (AID) chosen.

AID	ITPECHO Function
Enter	Echo
PF5	Automaticstring/repetition
PF6	Automatic string/repetition with definite response requested if allowed by the BIND parameters
PF9	Repeat last function
Clear	Restore the panel format
Logoff	Terminate the session

Note: All other attention identifiers will act like the attention identifier Enter.

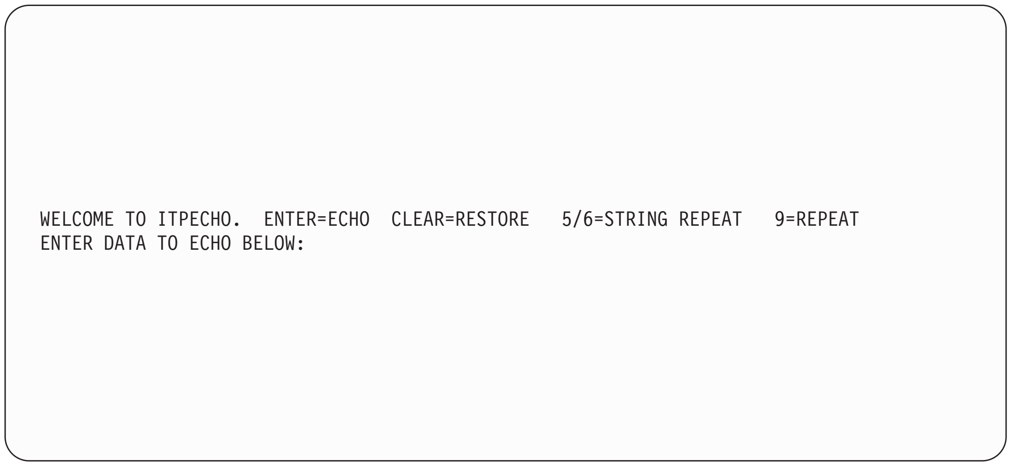
Enter—Echo function

The Enter AID requests that ITPECHO perform the standard echo function on the input data from the panel. Essentially, the 3270 fields are stripped out and the remaining content is sent back with a 3270 WRITE command.

ITPECHO assumes that the 3270 in session has at least a 24x80 panel display. After you log on, ITPECHO formats the panel into two logical fields, the top (protected) and the bottom (unprotected). The cursor is placed at the start of the unprotected input field. The protected field actually starts in row 24, column 80 of the lower right corner, and continues into the top half of the panel.

Figure 34 shows an example of the initial ITPECHO panel. When you enter data in the input field, the response will come back to the top of the panel.

Figure 34. Initial ITPECHO panel



```
WELCOME TO ITPECHO.  ENTER=ECHO  CLEAR=RESTORE   5/6=STRING REPEAT   9=REPEAT
ENTER DATA TO ECHO BELOW:
```

Type in the data to be echoed, starting at the initial cursor position. When you press the Enter key, ITPECHO expects to receive a data stream with the format of the incoming RU, as shown in Figure 35.

Figure 35. ITPECHO incoming data stream

AID	C	SBA	A	command	data

where:

AID	1-byte AID indicator
C	2-byte Cursor Position
SBA	1-byte Set Buffer Address command
A	2-byte address of input field that contains the following data:
command	ITPECHO 3270 command string
data	Incoming “data” echoed.

When ITPECHO receives an Enter AID, it strips the first six bytes from the RU, leaving a data portion. Then, ITPECHO places a 3270 WRITE command string into the output buffer. This string is followed by the data portion stripped off of the incoming RU. If the total length of the command string and data is longer than the allocated buffer size, the data is truncated to fit. You should be aware of this possible truncation when writing IF statements in message generation decks.

The WRITE command string causes the following events to be executed when the terminal receives the message:

1. The unprotected field is erased.
2. The protected field is erased up to, but not including, the middle rows that contain permanent operator information.
3. The data is written starting in row 1, column 1.
4. The cursor is again positioned at the input field, ready for another message.

If the original panel format was destroyed, such that the entire panel is one large input field, ITPECHO recognizes this and does not strip out too many characters in the input data. The panel format can be destroyed when the attribute bytes are overwritten, perhaps by a long (1920 bytes) message. In fact, the unformatted panel allows you to type more input data than the formatted panel. To restore the panel to its original form, simply press the Clear key.

PF5—Automatic string/repetition function

When you press PF5, an automatic string and repetition function is available. To use this feature, type in a set of integer request values on the terminal, starting at the initial cursor position. These values must conform to the following syntax:

```
LENG[,REPT[,INCR]] [DATA]
```

where:

LENG Specifies an integer from 0 to 32767. This requests an alphabetic string of this length to be sent to the terminal. LENG is required.

REPT Specifies an integer repetition factor from 1 to 32767. This requests that a total of *n* alphabetic string transmissions be sent in a row. The default is one transmission.

INCR Specifies an increment value from 0 to 32767. This requests that each subsequent string repetition be incremented by *n* characters in length. The default is zero.

DATA The data to be sent.

For example:

```
4,3,1 ABCDEF
```

These values request an initial string length of 4. The string is sent 3 times to the terminal; each time, the string length is incremented by one byte, as follows:

ABCD is sent.
ABCDE is sent.
ABCDEF is sent.

Notes:

- To use INCR, you must also use REPT.
- The first blank encountered in the request indicates the end of the request values.

- Any invalid specification causes an error message to be sent to the terminal.
- If the string length (including the command string) is too long to fit into the buffer, the string is truncated to fit.
- You can type in additional data following the request values, as long as it follows the delimiting blank character. (No blank is necessary if the INCR value contains four digits.) ITPECHO ignores this data and allows you, for example, to send a 30-byte message and receive a 100-byte message in response.
- ITPECHO will only process a PF5 request from a properly formatted panel. It will not process the function from an unformatted panel.
- When you ask ITPECHO to send more than 1130 bytes of data back to the simulated terminal (or a real terminal), the data will overlay the field definition with data and you may not be able to enter data. When you try to enter the data, you can get a ITP0403I message that indicates you are trying to enter data into a protected field. The following script lets you request more than 1130 bytes to be sent from ITPECHO to the simulated terminal and allows the simulated terminal to enter the next request.

```
0   IF LOC=D+0,TEXT=(ABC),THEN=CONT,SCAN=YES,STATUS=HOLD
    CURSOR ROW=22,COLUMN=1
    TEXT (1920 ),LENG=196
    PF5
*                              Use LCLEAR to do a local clear of
*                              the screen
    LCLEAR
    TEXT (xxx1920 ),LENG=196       Note xxx makes ITPECHO think
*                              there is a SBA
    PF5
    WAIT
```

  ITPECHO expects the first three bytes to be a Set Buffer Address (SBA). So you fool it with the “xxx”.

PF6—Automatic String/Repetition Function with Definite Response Requested

This PF key performs the same function as PF5, but also requests a definite response from the terminal if definite response is allowed by the BIND image for the session.

PF9—Repeat previous function

This function causes ITPECHO to transmit the last message sent to the terminal again. In the event that a number of messages were sent with the automatic string and repetition (PF5 or PF6) function, ITPECHO sends the entire sequence of messages again.

Clear—Clear and reformat the panel function

This function clears the display panel and causes ITPECHO to send the initial start-up message to initialize the panel. You can use Clear to restore the original panel format if it is accidentally (or purposely) destroyed.

Logoff—Terminate the session function

If you type in the logoff command at the terminal, ITPECHO recognizes the command and terminates the session. The proper format of the command is LOGOFF, and it must be the first string in the data. You can type in other data after the command, if at least one blank follows the word LOGOFF. The extra data is ignored.

Other AIDs—Echo function

Any other AID byte causes an echo function just as if the Enter key were pressed. If the AID carries data along with it after the cursor location, an echo will occur. If no data follows the AID, no data will be echoed (for example, PA 1, PA 2, or PA 3). See the previous discussion on the Enter key for more information on the echo function.

Functions available with non-3270 devices

Non-3270 devices (those that contained NON3270 as the INITIATE user data) do not receive special data stream considerations from ITPECHO. In fact, the only functions available in NON3270 operating mode are echo and logoff.

Echo

Any data received by ITPECHO (that does not cause a logoff) is sent back to the terminal in the exact format that it was received. No command insertion or modification is done. If the RU received is too long to fit into the buffer, ITPECHO truncates it to the length of the buffer.

Logoff

If the data stream received by ITPECHO contains LOGOFF (optionally followed by a blank and more data) starting at the first character, ITPECHO terminates the session.

Installing ITPECHO

You can install ITPECHO during the normal WSim installation procedure. Nothing additional needs to be done to install this program. However, some other preparation may be necessary before ITPECHO is run the first time. First, consider which APPL definition will be used by ITPECHO. Also, prepare your JCL procedure used to start ITPECHO, keeping in mind the ability to tailor the execution parameters.

For ITPECHO to communicate with VTAM, the application program must be defined with a VTAM APPL definition statement. The following example shows an APPL definition statement that defines an ITPECHO application program:

```
ITPECHO  APPL
```

You can use all of the default operand values for VTAM. You can optionally use password protection (PRTCT) in combination with the PASSWD execution parameter.

Although using the name ITPECHO could be the simplest choice, you can use any other application name. In fact, if extra unused APPL definitions exist in VTAM, you can use one of them without having to modify the data sets involved. You can then override the default APPLID execution parameter when running ITPECHO.

After adding any APPL definition statement, activate the application with the VTAM operator command VARY. The application must be active before ITPECHO can attempt to open the Access Method Control Block (ACB).

You must specify PARSESS=YES on the VTAM APPL definition when the WSim VTAM application has more than one active session, such as parallel sessions, with another VTAM application or logical unit.

Running ITPECHO

This section describes the execution parameters and the JCL and a TSO CLIST required to run ITPECHO.

Using ITPECHO execution parameters

You can enter the following execution parameters, which are optional, in the PARM field for the JCL EXEC statement or for the TSO CLIST CALL statement when you run ITPECHO.

APPLID=*name*
: Specifies the VTAM application identifier for the ACB. The *name* can be any 1- to 8-character alphanumeric name that VTAM will accept as a valid application identifier. If not coded, the default APPLID is ITPECHO.

BUFSIZE=*nnnnn*
: Specifies a buffer size to be used for each input and output buffer associated with each session. These two buffers are allocated when a session is requested and will limit maximum message lengths to the specified size. The number of bytes in the buffer *nnnnn* is an integer from 20 to 32767. If not coded, the default BUFSIZE is 2048.

PASSWD=*password*
: Specifies the *password* to be used when opening the ACB with VTAM. This should match the PRTCT operand value on the VTAM APPL definition statement for the ACB that is being used. If no PRTCT operand is coded in VTAM, no password checking is done and any PASSWD value will be accepted.

  The *password* can be any 0- to 8-character alphanumeric password that VTAM will accept as a valid password. If not coded, the default PASSWD is left blank. Coding PASSWD with no value resets the password value to blanks.

SMSG
: Causes session initialization and termination messages to be written to the operator with a WTO. This is the initial setting for ITPECHO.

NOSMSG
: Suppresses session initialization and termination messages. This can be useful if hundreds of sessions are involved during the run.

TRACE
: Starts an internal trace within ITPECHO. This is mainly for debugging purposes and may be required for problem determination.

NOTRACE
: Stops the internal trace if it was active. This is the initial setting for ITPECHO.

WTOR
: Causes the WTORs associated with messages ITP904E and ITP905E to be issued. This is the initial setting for ITPECHO.

NOWTOR
: Suppresses the WTORs associated with messages ITP904E and ITP905E. NOWTOR allows ITPECHO to be executed without operator intervention.

GNAME = *generic_resource_name*
: Allows ITPECHO to associate itself with the generic resource name specified.

After ITPECHO is started, you can change the execution parameters before the ACB is opened if the WTOR execution parameter is specified or defaulted. The current parameters will be displayed with message ITP900I, and you can reply to message ITP904E with new execution parameters (in the format listed above), U to use the current settings, or END to terminate program execution. Refer to *WSim Messages and Codes* for more information about these messages.

If you enter new parameters, they are displayed again, and you are prompted for additional changes, U, or END. If you made an error in the specification of any parameter, the entire parameter list is set to the defaults. Once you reply U to the message, the ACB is opened and ITPECHO is ready to accept logon requests.

Using JCL

Below shows an example of JCL you can use to run ITPECHO on MVS.

```
//ITPECHO  PROC
//JOBLIB   DD  DSN=WSIM.SITPLOAD
//ITPECHO  EXEC PGM=ITPECHO,
//          PARM=('APPLID=YOURAPPL,BUFSIZE=2000',
//          'PASSWD=APPLPASS')
//
```

Using a TSO CLIST

The following example is a CLIST you can use to run ITPECHO under TSO.

```
CALL 'WSIM.SITPLOAD(ITPECHO)'
```

Note: If you run ITPECHO under TSO, you cannot use the ID or terminal while the job is running.

Using ITPECHO operator commands

You can start ITPECHO as a normal system job by using the MVS Start command or by submitting the JCL. Once ITPECHO is started, message ITP905E leaves an outstanding reply for operator commands if the WTOR execution parameter is specified or defaulted. In response to this message, the operator commands listed in the following section may be issued while ITPECHO is running. Only one command can be entered at a time.

SMSG	Causes session initialization and termination messages to be written to the operator with a WTO. This is the default for ITPECHO.
NOSMSG	Suppresses session initialization and termination messages. This is useful if hundreds of sessions are involved during the run.
TRACE	Starts an internal trace within ITPECHO. This is mainly for debugging purposes, and may be required for problem determination.
NOTRACE	Stops the internal trace if it was active.
END	Causes ITPECHO to terminate. This is the normal method of ending the program.

Logging on to ITPECHO

Logging on to ITPECHO is a simple process if you understand the different procedures and the SNA protocols used during session initiation.

There are three components of a logon request that need special consideration when using ITPECHO:

- The application identifier. The application identifier is always required for any logon request.

- The optional INITIATE_SELF user data. The user data field of the INITIATE_SELF is applicable only for non-3270 devices that will use ITPECHO. If this field is NON3270, no 3270 data streams are used by ITPECHO. Otherwise, 3270 data stream format is assumed in transmit and receive messages.
- The logmode (BIND image). The logmode name is optional, but should indicate an LU2 BIND image for WSim LU2 devices.

You can use one of the following common methods to initiate a session with the application and a terminal (simulated or real):

1. V NET,LOGON=ITPECHO,ID=*luname*

   The system operator can use the VARY LOGON VTAM command to initiate a session with the application and the terminal (or group of terminals). Refer to *ACF/VTAM Programmer's Guide* for more information on this command. With this method, you can specify a LOGMODE operand but user data (such as NON3270) is not allowed.
2. LOGON APPLID(ITPECHO)

   The terminal operator can issue the logon command from the terminal in SSCP-LU session to initiate a session with ITPECHO. Both LOGMODE and DATA operands are available on this command. A sample logon request for a non-3270 device might be:
   ```
   LOGON APPLID(ITPECHO) LOGMODE(WSIMLU0) DATA(NON3270)
   ```
3. SNA INITIATE_SELF Request

   An SNA INITIATE_SELF request may be generated by the WSim terminals, especially for cross-domain simulation. The RESOURCE would be ITPECHO, and again, BIND images and user data may also be included depending on the device.

   From a WSim script, the CMND statement could be coded like this:
   ```
   CMND COMMAND=INIT,RESOURCE=ITPECHO,MODE=WSIMLU0,DATA=(NON3270)
   ```
4. VTAMLST Definitions

   VTAMLST definition statements for the logical units have a variety of operands allowing them to be assigned to a specific application permanently. This method of initiating sessions is easy and it eliminates some of the logon scripting that must be done in WSim.

For information on running the sample installation network supplied with WSim, refer to *WSim Script Guide and Reference*.

Understanding ITPECHO return codes

After running, ITPECHO sets a return code to indicate the status of the execution. ITPECHO can return the following codes:

Code	Meaning
0	Execution completed successfully.
4	The program ended during parameter specification.
8	Storage was not available for initialization.
12	The ACB could not be opened with VTAM.

Chapter 11. Simulated resource type codes

This chapter lists the code values for each simulated resource type. You can use the codes from this table to identify resource types when reading a log data set listing or processing in a user exit routine.

The following lists the code values for terminal resource types:

Terminal	Type
TCP/IP	30
VTAMAPPL	69

The following lists the code values for device resource types:

Device	Type	Device	Type
FTP (command conn)	91	LU0	E0
FTPD (data conn)	92	LU1	E1
STCP	93	LU2	E2
TN3270	94	LU3	E3
TN3270E	95	LU4	E4
TN3270P	96	LU6	E6
SUDP	97	LU7	E7
TNNVT	98	LU6.2	E9
TN5250	99		

Chapter 12. Understanding message logging

This chapter describes message logging, time stamping, writing messages to the log data set, and logging messages under specific conditions, including:

- File Transfer Protocol (FTP) over a TCP/IP network
- CPI-C

What is message logging?

The WSim message logging facility, when active, writes messages to the log data set containing all data that WSim simulated resources transmit or receive in a specified network. Most WSim users use the message logging facility because of its usefulness for analyzing network simulations.

You define the name of the data sets that will be used for message logging in the LOGDD DD statements when you run WSim. By default, the message logging facility is active for the entire network. You can override the default in your network definition by specifying MLOG=NO on the NTWRK, or other network definition statements.

Furthermore, you can code the NTWRKLOG statement when you define a network to specify that a separate log data set be used for that network. This enables you to run multiple networks and analyze the results from each network independently at a later time.

Each record on the log data set contains an 88-byte header, followed by the data transmitted or received, an informational message, or trace data. You use the MLEN operand on the NTWRK or other network definition statements to specify the maximum length of the data portion of a record. Each record belongs to one of the following record types:

Console record (CNSL)
Contains either an operator command or an operator command response in the data portion of the record. Operator commands are always logged.

CPI-C trace record (CTRC)
Contains general messages that trace the execution of CPI-C transaction programs (TPs).

Informational record (INFO)
Is written to the log data set when errors occur during a simulation run, when message generation starts or ends, or when a user exit routine invokes the WSim interface routine for logging data.

Log record (LOG)
Is written to the log data set whenever a LOG statement or a LOG operand on an IF statement is encountered during message generation. The data is written in an interpreted dump format so that it appears in hexadecimal and EBCDIC notation.

Log display record (DSPY)
Is written to the log data set each time a simulated 3270 or 5250 display or printer image buffer is written to the log data set as a result of the LOGDSPLY operand or the LOG DISPLAY statement.

Marker record (MARK)
Is written to the log data set each minute of the simulation run. This record contains a header only; it contains no data.

Message data record (XMIT and RECV)
Contains the data generated or received by a WSim-simulated resource.

Message trace record (MTRC)
Contains general messages about the message generation path through decks (EVENTS) as well as the IF messages about logic tests.

STL trace record (STRC)
Contains general messages about the execution of an STL program.

Verify data record (VRFY)
Contains information relevant to IF statements processed with the VERIFY action specified.

For more information about the different log record types, refer to the list of record types in "Log record header" on page 33. The following sections discuss how messages are time stamped and how data messages that are received or transmitted by a simulated resource are logged.

How messages are time stamped

Each record written to the WSim log data set contains three time stamps in the header portion of the record. These time stamps are labeled START, STOP, and READY. The information contained in each time stamp field depends on whether the record is a message data record.

Time stamps for records that are not message data records

Log data set records that are not message data records (for example, informational, marker, console and display records) contain the following data in the time stamp fields of the record header:

START field
Contains a time stamp that gives the time that the record was created in the WSim host processor.

STOP field
Contains the date as packed decimal digits of the form:

```
0CYYDDDF
```

where:

C is a digit representing centuries beyond the twentieth. In the years 1900 through 1999, C has a value of 0. In the years 2000 through 2099, C has a value of 1.

YY is the last two digits of the year

DDD is the day of the year

F is a 4-bit sign character

For example, January 21, 2002 would be returned as `0102021F`.

READY field
Contains the WSim release level left-justified in the field.

Time stamps for message data records

The information contained in the time stamp fields of the record header for message data records varies depending on the following:

- Whether the data was transmitted or received
- The type of simulation.

The following sections define the START, STOP, and READY time stamps for several types of simulation. Note that for a data message transmitted by a simulated resource, the READY time stamp is created in the WSim host processor when the resource has completed its “think” time and is ready to transmit a message in response to the next poll received. The START and STOP time stamps are set differently depending on the type of simulation being performed.

VTAMAPPL simulation

For a data message transmitted from a simulated resource to the system under test using the VTAM API, the START time stamp is created in the WSim host processor when the send request is queued to VTAM. The STOP time stamp is created in the WSim host processor when WSim receives notification from VTAM that the send request is complete.

For a data message received by a simulated resource using the VTAM API, the START time stamp is created in the WSim host processor when WSim receives notification from VTAM that data has been received. The STOP and READY time stamps are equal and are created in the WSim host processor when WSim receives notification from VTAM that all the data has been moved into a receive buffer.

CPI-C transaction program simulation

For a data message transmitted from a simulated CPI-C transaction program to the system under test, the START time stamp is created in the WSim host processor when the send request is queued to VTAM. The STOP time stamp is created in the WSim host processor when WSim receives notification from VTAM that the send request is complete.

For a data message received by a simulated CPI-C transaction program, the START time stamp is created in the WSim host processor when WSim receives notification from VTAM that data has been received. The STOP and READY time stamps are equal and are created in the WSim host processor when the WSim script issues a CPI-C verb that retrieves the received data.

TCP/IP client simulation

For a data message transmitted from a simulated resource to the system under test using a TCP/IP connection, the START time stamp is created in the WSim host processor when the send() socket call is issued. The STOP time stamp is created in the WSim host processor when WSim receives notification from TCP/IP that the send request is complete.

For a data message received by a simulated resource using a TCP/IP connection, the START time stamp is created in the WSim host processor when the recv() socket call is issued. The STOP and READY time stamps are equal and are created in the WSim host processor when WSim receives notification from TCP/IP that all the data has been moved into a receive buffer.

How data messages are logged

In general, messages to or from WSim-simulated resources are written to the log data set immediately after they are sent or received. The messages are logged in the same format in which they are sent or received, with the following exceptions:

- The logging of File Transfer Protocol (FTP) command data and FTP file data.

Each of these special cases is discussed in more detail in the following sections.

Logging FTP command data and FTP file data

Because FTP command data and FTP file data flow on separate TCP/IP connections, the connection on which each unit of data flows is identified in the WSim log data set and indicated in the formatted output by ITPLL. When WSim logs ASCII data on the data connection, the interpreted portion of the formatted output shows the ASCII characters represented by the hexadecimal data. WSim always logs data flowing on the command connection in EBCDIC, even though the data actually flows in ASCII.

WSim flags data not actually transmitted to or received from the FTP server with a "%" character prior to the XMIT or RECV in the log header.

CPI-C transaction program message logging

A data message for a CPI-C transaction program is any data that flows to or from a conversation partner. This data may be in the form of data sent or received, attach requests (FMH-5s) sent or received, or error log data sent by the transaction program.

Data, attach requests, and error log data that is sent from the simulated transaction program to a conversation partner is logged using the XMIT record type. Data and attach requests that are received from a conversation partner by the simulated transaction program are logged using the RECV record type. As with other simulation types, the transmit and receive records will only be present in the WSim log if message logging was turned on(MLOG=YES) when the simulation was run.

Chapter 13. Using the TCP/IP Trace Utility

The TCP/IP Trace Utility uses the real-time application-controlled TCP/IP trace Network Management Interface (NMI) to capture TCP/IP data trace records. TCP/IP data trace records contain the data that is exchanged between a server and a client. The utility saves the trace records to a data set that the ITPIPGEN program can process. An STL program can then be generated from TCP/IP trace records.

Running the TCP/IP Trace Utility

The TCP/IP Trace Utility requires information to create the trace. You can supply the information in one of the following ways:

- Enter it in fields on the WSim/ISPF Interface panel for the TCP/IP Trace Utility.
- Offer it as execution parameters to the trace utility program ITPIPTRX.

The JCL that you use to run the trace utility provides the following locations:

- The data set that is used to save the TCP/IP data trace records
- The printer

To use the TCP/IP Trace Utility, you must have READ access to the following SAF resource profiles in the SERVAUTH class:

- EZB.TRCCTL.*sysname.tcpname*.OPEN
- EZB.TRCCTL.*sysname.tcpname*.DATTRACE

Where:

- *sysname* is the MVS system name where the TCP/IP stack is running
- *tcpname* is the TCP/IP stack job name

To use the TCP/IP Trace Utility to obtain a trace in clear text for a server that uses AT-TLS for message encryption, you must have READ access to the following SAF resource profile in the SERVAUTH class: EZB.TRCSEC.*sysname.tcpname*.AT-TLS

Note: When simulating TCP/IP clients, WSim does not have support for establishing a secure connection by using SSL. Therefore, when you use WSim to simulate a client that requires a secure connection, you must define a TCP/IP networking policy to ensure that the simulated client uses AT-TLS.

For further information on the SAF resources, see *z/OS® Communications Server: IP Programmer's Guide and Reference*.

For further information on TCP/IP networking policies, see *z/OS Communications Server: IP Configuration Guide* and *z/OS Communications Server: IP Configuration Reference*.

The following sections describe more information about the TCP/IP Trace Utility:

- Running the TCP/IP Trace Utility using the WSim/ISPF Interface
- Execution parameters
- An example of JCL

Note: The TCP/IP Trace Utility cannot be run via a TSO REXX/CLIST.

Using the WSim/ISPF Interface

You can run the TCP/IP Trace Utility from the WSim/ISPF Interface. To do this, follow these steps:

1. Invoke the WSim/ISPF Interface main panel from ISPF. The method that you use to do this depends on how the WSim/ISPF Interface application is installed at your site. If you are not sure how to do this, see your system programmer for assistance.
2. Select option 4 from the WSim/ISPF Interface main panel and press **Enter**. The TCP/IP Trace Processing menu is displayed.

   Note: You can also type "TCPPROC" on the command line and press **Enter** to display this panel.
3. Select option 1 from the TCP/IP Trace Processing menu and press **Enter**. The Generate a TCP/IP Data Trace panel is displayed.

   Note: You can also type "TCPTRC" on the command line and press **Enter** to display this panel.
4. Fill in the appropriate information on this panel and press **Enter** to run the TCP/IP Trace Utility.

For more information on the WSim/ISPF Interface, see Chapter 2, "Running WSim with the WSim/ISPF Interface," on page 5.

Using TCP/IP Trace Utility execution parameters

You can enter the following execution parameters in the PARM field for the JCL EXEC statement or on the CALL statement for TSO CLISTs when you run the TCP/IP Trace Utility.

STACK=*name*
Specifies the name of the z/OS TCP/IP stack that handles the communication between the server and client.

PORT=*nnnnn*
Specifies the port that is used by the server that executes on z/OS.

IP=*ipaddr*
Specifies the IP address of the client that communicates with the server. The IP address can be either an IPV4 address or an IPV6 address.

IDLE=*nnnnn*
Specifies an idle time limit value in seconds. Idle time occurs when there is no communication between the server and client. If idle time exceeds the specified limit, WSIM stops the trace. The default idle time limit is 180 seconds.

Note: Whenever a message is exchanged between the server and client, the idle time value is reset to zero.

PRTLNCNT=*nnn*
Specifies the maximum number of lines to be printed on a page of output before ejecting to a new page. The value for *nnn* is an integer from 35 to 255. The default value for *nnn* is 60.

Using JCL

The following JCL statements are required to run the TCP/IP Trace Utility on MVS.

Statement	Function
TRJOB1 JOB	Initiates the job.
DELETE EXEC	Invokes program IEFBR14 to enable the deletion of the output trace data set if it exists.
DELFILE DD	Specifies the data set where the TCP/IP trace records are saved.
TCPDATRC EXEC	Specifies the TCP/IP Trace Utility program ITPIPTRX.
STEPLIB DD	Defines the data set that contains the WSim host processor modules.
TCPTRACE DD	Defines the data set that is used to save the TCP/IP trace records.
SYSPRINT DD	Defines the output printer. SYSPRINT records can be either fixed or variable length. For fixed-length records, logical record lengths of 133 to 256 are accepted. For variable-length records, logical record lengths of 137 to 260 are accepted. In either case, the maximum length of non-control printed data is 255 bytes. The default is fixed 133-byte length with blocking supported.

The following example shows the JCL that you can use to run the TCP/IP Trace Utility.

```
//TRJOB1 JOB (1234,1234),'VANDYKE',MSGCLASS=X,
//         CLASS=A,NOTIFY=&SYSUID
//*
//DELETE   EXEC PGM=IEFBR14
//DELFILE  DD DSN=TCP.DATA.TRACE,DISP=(MOD,DELETE),
//         SPACE=(TRK,(0))
//TCPDATRC EXEC PGM=ITPIPTRX,
//            PARM=('STACK=TCPIP,IP=9.190.124.72,',
//            'PORT=6003,IDLE=180')
//STEPLIB  DD DSN=WSIM.SITPLOAD,DISP=SHR
//TCPTRACE DD DSN=TCP.DATA.TRACE,
//            DISP=(NEW,CATLG),UNIT=SYSALLDA,
//            DCB=(RECFM=VB,LRECL=27994,BLKSIZE=27998),
//            SPACE=(CYL,(1,1))
//SYSPRINT DD SYSOUT=A
```

Understanding TCP/IP Trace Utility return codes

The TCP/IP Trace Utility provides a return code to indicate the status of the execution. The TCP/IP Trace Utility can return the following codes:

Code	Meaning
0	The run was completed with no errors.
1004	An invalid or unknown parameter was specified.

Code	Meaning
1005	Storage was not available for TCP/IP Trace Utility execution.
1008	The load of a WSim module failed.
1010	The ATTACH of the trace interface module ITPIPTRC failed.
1012	The open of the WSim load library as a task library failed.
1014	The SYSPRINT data set failed to open.
1015	The creation of a name or a token for this trace instance failed.
1018	A request to start a new trace was invalid because there was already a trace active for this TSO/ISPF user.
1020	A STOP or QUERY request was received but no TCP/IP trace is active.
1022	An output trace data set was not allocated to the TCPTRACE DD.
1024	The data set type for the output trace data set was invalid.
1025	Data set concatenation was invalid for the TCPTRACE DD.
1030	The TCP/IP data trace task failed to process the trace request.
1031	The TCP/IP stack name specified was not known to TCP/IP.
1032	The trace failed due to an error returned by the TCP/IP real-time application controlled trace interface.
1035	The open of the TCP/IP data trace output data set failed.

Chapter 14. Using the TCP/IP Trace Formatting Utility

The TCP/IP Trace Formatting Utility produces a formatted report of the TCP/IP trace records that are saved in a data set. The utility calls TCP/IP trace record formatting Network Management Interface (NMI) to handle the formatting of trace records.

The following sections describe more information about the TCP/IP Trace Formatting Utility:

- Running the TCP/IP Trace Formatting Utility using the WSim/ISPF Interface
- Examples of JCL and a TSO CLIST

Running the TCP/IP Trace Formatting Utility

The TCP/IP Trace Formatting Utility requires as input a data set that contains TCP/IP trace records. The JCL or TSO REXX/CLIST you use to run the formatting utility provides the following locations:

- TCP/IP trace data set
- The printer

Using the WSim/ISPF Interface

You can run the TCP/IP Trace Formatting Utility from the WSim/ISPF Interface. To do this, follow these steps:

1. Invoke the WSim/ISPF Interface main panel from ISPF. The method that you use to do this depends on how the WSim/ISPF Interface application is installed at your site. If you are not sure how to do this, see your system programmer for assistance.
2. Select option 4 from the WSim/ISPF Interface main panel and press **Enter**. The TCP/IP Trace Processing menu is displayed.

   Note: You can also type “TCPPROC” on the command line and press **Enter** to display this panel.
3. Select option 2 from the TCP/IP Trace Processing menu and press **Enter**. The Format a TCP/IP Data Trace panel is displayed.

   Note: You can also type “TCPFMT” on the command line and press **Enter** to display this panel.
4. Fill in the appropriate information on this panel and press **Enter** to run the TCP/IP Trace Formatting Utility.

For more information on the WSim/ISPF Interface, see Chapter 2, “Running WSim with the WSim/ISPF Interface,” on page 5

Using JCL

The following JCL statements are required to run the TCP/IP Trace Formatting Utility on MVS.

Statement	Function
FTJOB1 JOB	Initiates the job.
STEP1 EXEC	Specifies the program name.
STEPLIB DD	Defines the data set that contains the WSim host processor modules.
TRACIN DD	Specifies the data set that contains the TCP/IP trace records.
SYSPRINT DD	Defines the output printer. SYSPRINT records can be either fixed or variable length. For fixed-length records, logical record lengths of 133 to 256 are accepted. For variable-length records, logical record lengths of 137 to 260 are accepted. In either case, the maximum length of non-control printed data is 255 bytes. The default is fixed 133-byte length with blocking supported.

The following example shows an example of JCL that you can use to run the TCP/IP Trace Formatting Utility.

```
//FTJOB1 JOB
//*
//STEP1    EXEC PGM=ITPIPFMT
//STEPLIB  DD DSN=WSIM.SITPLOAD,DISP=SHR
//TRACIN   DD DSN=TCP.DATA.TRACE,DISP=SHR
//SYSPRINT DD SYSOUT=A
```

Using a TSO CLIST

The following example shows the TSO CLIST that you can use to run the TCP/IP Trace Formatting Utility.

```
ALLOC DDNAME(SYSPRINT) SYSOUT(A)
ALLOC DDNAME(TRACIN) DSNAME('TCP.DATA.TRACE') SHR
CALL  'WSIM.SITPLOAD(ITPIPFMT)'
FREE DDNAME(SYSPRINT)
FREE DDNAME(TRACIN)
```

Understanding TCP/IP Trace Formatting Utility return codes

The TCP/IP Trace Formatting Utility provides a return code to indicate the status of the execution. The TCP/IP Trace Formatting Utility can return the following codes:

Header	Header
0	The run was completed with no errors.
8	Storage was not available for TCP/IP Trace Formatting Utility execution.
10	The TCP/IP trace data set failed to open.
12	The load of TCP/IP trace formatting stub module EZBNMCTF failed.
16	The SETUP call to the TCP/IP trace formatting interface EZBCTAPI failed.
18	An invalid TCP/IP trace record was found.

Part 2. Script generating utilities

Chapter 15. Generating scripts interactively with IDC

The Interactive Data Capture (IDC) Utility ITPIDC provides a simple and easy-to-use method of creating WSim scripts for simulated 3270 devices. IDC is a stand-alone VTAM application program that can run under MVS. To create a WSim script, you simply log on to an application through IDC from a real 3270 display and perform the actions you want WSim to simulate. IDC records the interactions between the display and the application and stores the captured SNA traffic in its own log data set. No VTAM buffer or NPM VTAMLOG traces are needed with IDC. IDC supports one 3270 display user (single session) at a time for each copy of the IDC program executing.

The log created by IDC is compatible with the log created by a WSim simulation run. As a result, you can format and print the IDC log with the Loglist Utility or use it as input to the Response Time Utility. You can also compare the IDC log with a WSim log from a simulation run to verify test results with the Log Compare Utility.

From the log, IDC can create either a Structured Translator Language (STL) program or a WSim Scripting Language message generation deck. Choosing the Type of Script to Generate discusses the advantages and disadvantages of each type.

From an SNA perspective, communication between a real 3270 device and a host application program through IDC involves two sessions. The first session is between IDC and the 3270 display. You establish this session when you log on to IDC. IDC is the Primary Logical Unit (PLU) and the 3270 display is the Secondary Logical Unit (SLU). The second session is between IDC and the host application. You establish this session when you request a session with the host application through IDC. In this session, IDC is the SLU and the host application is the PLU.

IDC acts as a bridge between these two sessions and passes data back and forth between the 3270 display and the host application. This is shown in Figure 36.

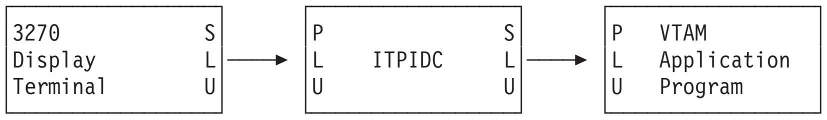


Figure 36. Interactive Data Capture session flow

IDC supports 3270 LU type 2 and LU type 0 SNA sessions.

Creating scripts with IDC involves the following tasks:
1. Setting up IDC
2. Starting IDC
3. Establishing sessions
4. Capturing data
5. Generating scripts
6. Stopping IDC
7. Modifying IDC generated scripts (optional)

8. Debugging problems between IDC, users, and applications (optional).

After you use IDC to generate a script, you need to create a network definition (see Creating Network Definitions). WSim uses both the network definition and script during the network simulation run. The network definition tells WSim what devices to simulate and the script tells WSim what data those devices should send.

Before you begin, carefully plan what you want to do and what data you want to capture to turn into WSim scripts. For information about planning a WSim test, see *WSim User's Guide*.

Setting up IDC

WSim must first be installed to use IDC. See *WSim User's Guide* for information about installing WSim.

You must perform the following steps to set up IDC:
1. Define IDC to VTAM
2. Allocate data sets for IDC.

Defining IDC to VTAM

To define IDC to VTAM, code a VTAM APPL definition statement under a VTAM application program major node. For example:

```
ITPIDC   APPL
```

The name coded on the APPL statement (ITPIDC in the example) can be any valid VTAM APPL name. You need no additional operands for IDC. However, the MODETAB= operand may be required to access the logon mode table used at your installation. If you are running IDC from the WSim/ISPF Interface or executing IDC with the TSOCON execution parameter, coding SESSLIM=YES on the APPL statement is recommended since it forces a session limit of one for the IDC application which is consistent with a real 3270 display. Do not code SESSLIM=YES if you will not be running IDC from the WSim/ISPF Interface and the TSOCON execution parameter willnot be used. Refer to *VTAM Installation and Resource Definition* for details.

The APPL name you select depends on the requirements of the applications or subsystems that you test. For example, to test a CICS® system that uses the auto-install facility, choose an APPL name that conforms to CICS naming conventions.

Multiple VTAM APPL definitions must be defined to support more than one copy of IDC executing at the same time.

Allocating IDC data sets

To run IDC under MVS, you need to allocate or use an existing allocation for the following data sets:

IDC log
: This data set contains the captured data. The space needed for this data set depends on the number of transactions and amount of data you are capturing. Initially, allocate at least 150 blocks (5 cylinders of 3380 DASD or equivalent) for this data set. For 3380 DASD, allocate this data set as a variable block, sequential or partitioned data set with a record length of 23472 bytes and a block size of 23476 bytes. For 3390 DASD, allocate this data set as a variable

block, sequential or partitioned data set with a record length of 27994 bytes and a block size of 27998 bytes. For other DASD types, the block size should be the largest value (less than or equal to 32760) that best uses the space available on each track. The record length value must be 4 bytes less than the block size value. This data set must be cataloged.

You may want to allocate multiple log data sets or allocate a partitioned log data set depending on how you plan to use IDC. See "Changing log data sets" on page 188 for a discussion of the use of multiple log data sets.

STL programs

This data set contains the generated STL programs. If you plan to generate STL programs, allocate this as a fixed block, variable block, or variable data set with a record length of at least 71 bytes and a block size compatible with the record length. You may allocate this as either a partitioned or sequential data set. The space needed for this data set depends on the number of user actions and amount of data generated in the program. Initially, allocate at least 5 cylinders of 3380 DASD or equivalent space for this data set. This data set must be cataloged.

You can also use existing data sets you normally use for STL programs.

WSim message generation decks

This data set contains the generated WSim Scripting Language message generation decks. If you plan to generate message generation decks, allocate this as a fixed block data set with a record length of 80 bytes and a block size compatible with the record length. You may allocate this as either a partitioned or sequential data set. The space needed for this data set depends on the number of user actions and amount of data generated in the message generation decks. Initially, allocate at least 5 cylinders of 3380 DASD or equivalent for this data set. This data set must be cataloged.

You can also use existing data sets you normally use for WSim message generation decks.

IDC defaults

This data set contains user-defined default values for IDC panel entry fields. Allocate a fixed block, sequential data set with a record length of 2087 bytes and a block size of 2087 bytes. You only need to allocate 1 block for this data set.

IDC trace

This is an optional data set and is only needed for debugging communication problems between IDC and displays or applications. (See "Analyzing the IDC trace" on page 207 for more information.) The space needed for this data set depends on the number of transactions and amount of data you are capturing. Initially, allocate at least 150 blocks (5 cylinders of 3380 DASD or equivalent) for this data set. For 3380 DASD, allocate this data set as a variable block, sequential data set with a record length of 23472 bytes and a block size of 23476 bytes. For 3390 DASD, allocate this data set as a variable block, sequential data set with a record length of 27994 bytes and a block size of 27998 bytes. For other DASD types, the block size should be the largest value (less than or equal to 32760) that best uses the space available on each track. The record length value must be 4 bytes less than the block size value.

Remember, the sizes given above are just estimates to get you safely started. You should base any long range predictions for space requirements on expected use and experience with IDC.

Note: IDC does not support tape data sets.

Starting IDC

Once you set up IDC, you then can start the IDC utility. IDC runs as a stand-alone VTAM application program. You do not need to run WSim to use IDC. IDC supports only a single user at a time. If multiple users want to use IDC at the same time, each must have a separate copy of IDC.

Defining the IDC job stream

Under MVS, you can run IDC as a batch job, as a started procedure, as a CLIST or EXEC under TSO, or by way of the WSim/ISPF Interface.

Running IDC from the WSim/ISPF Interface

To invoke IDC from the WSim/ISPF Interface, follow these steps:

1. Invoke the WSim/ISPF Interface main panel from ISPF. The method you use to do this depends on how the WSim/ISPF Interface application is installed at your site. If you are not sure how to do this, see your system programmer for assistance.
2. Select option 3 from the WSim/ISPF Interface main panel and press **Enter**. The Interactive Capture and Build Message Decks and STL Programs panel is displayed.

   Note: You can also type "IDC" on the WSim/ISPF Interface main panel command line and press **Enter** to display this panel.
3. Fill in the appropriate fields on this panel and press **Enter** to run IDC.

For more information on the WSim/ISPF Interface, refer to Part 1, "General utilities," on page 1.

Running IDC as an MVS batch job

Below shows the JCL to run IDC as an MVS batch job. Note that the optional execution parameters "NOWTOR,NOWTO" are coded to suppress all IDC console messages. You need to use option 4 (End IDC) from the IDC Main panel (Figure 37 on page 182) to end the IDC utility with this example.

```
//IDC       JOB
//ITPIDC    EXEC     PGM=ITPIDC,PARM='NOWTOR,NOWTO'
//STEPLIB   DD       DSN=WSIM.SITPLOAD,DISP=SHR
//SYSPRINT  DD       SYSOUT=A
//SYSUT1    DD       UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//IDCDFLTS  DD       DSN=WSIM.IDCDFLTS,DISP=OLD
```

If you code the "TRACE" execution parameter, add the following line:

```
//IDCTRACE DD       DSN=WSIM.IDCTRACE,DISP=OLD
```

Running IDC as an MVS started procedure

The JCL to run IDC as an MVS started procedure is shown in the example below. You need to place the JCL in SYS1.PROCLIB or the appropriate data set for your system. To end the IDC utility, you need to use option 4 (End IDC) from the IDC Main panel (Figure 37 on page 182) or issue the MVS stop (P IDC) or modify (F) END command.

```
//IDC       PROC
//ITPIDC    EXEC     PGM=ITPIDC
//STEPLIB   DD       DSN=WSIM.SITPLOAD,DISP=SHR
```

```
//SYSPRINT DD       SYSOUT=A
//SYSUT1   DD       UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//IDCDFLTS DD       DSN=WSIM.IDCDFLTS,DISP=OLD
```

If you code the “TRACE” execution parameter, add the following line:

```
//IDCTRACE DD       DSN=WSIM.IDCTRACE,DISP=OLD
```

Running IDC from a CLIST

The example below shows the CLIST commands to run IDC under TSO without the TSOCON execution parameter.

```
ALLOC    DDNAME(SYSPRINT) SYSOUT(A)
ALLOC    DDNAME(SYSUT1)   UNIT(SYSALLDA) SPACE(1,1) CYL
ALLOC    DDNAME(IDCDFLTS) DATASET('WSIM.IDCDFLTS') OLD
CALL     'WSIM.SITPLOAD(ITPIDC)'
FREE     DDNAME(SYSPRINT SYSUT1 IDCDFLTS)
```

The following example shows the CLIST commands to run IDC under TSO with the TSOCON execution parameter.

```
CONTROL NOMSG
FREE     DDNAME(SYSPRINT SYSUT1 IDCDFLTS IDCTRACE)
CONTROL MSG
ALLOC    DDNAME(SYSPRINT) SYSOUT(A)
ALLOC    DDNAME(SYSUT1)   UNIT(SYSALLDA) SPACE(1,1) CYL
ALLOC    DDNAME(IDCDFLTS) DATASET('WSIM.IDCDFLTS') OLD
SET SYSOUTTRAP = 2
PROFILE
DO &I = 1 TO &SYSOUTTRAP
  SET STRING = &&SYSOUTLINE&I
  SET INDEX = &SYSINDEX(NOINTERCOM,&STRING)
  IF &INDEX > 0 THEN +
    SET NOINTERCOM = 1
END
IF &NOINTERCOM ¬= 1 THEN +
  PROFILE NOINTERCOM
SET SYSOUTTRAP = 0
CALL 'WSIM.SITPLOAD(ITPIDC)' 'TSOCON'
SET RC = &LASTRC
IF &RC > 0 THEN +
  DO
    WRITE IDC ended with return code &RC.
    WRITE Refer to the IDC log data set for error messages.
  END
IF &NOINTERCOM ¬= 1 THEN +
  PROFILE INTERCOM
CONTROL NOMSG
FREE     DDNAME(SYSPRINT SYSUT1 IDCDFLTS)
CONTROL MSG
EXIT
```

If you code the “TRACE” execution parameter, add an ALLOCATE and a FREE statement for the IDCTRACE data set.

```
ALLOC    DDNAME(IDCTRACE) DATASET('WSIM.IDCTRACE') OLD
⋮
FREE     DDNAME(IDCTRACE)
```

Specifying execution parameters

You can specify the following optional execution parameters when you run IDC:

APPLID=*name*
: Specifies the VTAM application identifier for the Access Method Control Block (ACB). *name* can be any 1 to 8 character alphanumeric name that VTAM

accepts as a valid application identifier. This is the name coded on the VTAM APPL statement for IDC. If you do not specify a value for *name*, IDC uses ITPIDC as the default value for this execution parameter.

CKBNDQRY
Causes IDC to check the 3270 Query Reply bit in the BIND (byte 15, bit 0). If the bit is on, IDC sends a 3270 query to the device to determine what functions the display supports. If the bit is off, IDC does not send the query. If you do not code this parameter, IDC will always send the query.

You would code this parameter in situations where an exception response, generated by a display not supporting 3270 Query, would cause the session with IDC to be aborted without allowing IDC to recover. This situation sometimes occurs when intermediate VTAM applications are between the 3270 display and IDC.

DEBUG
Causes comments to be inserted into the generated scripts before each group of generated statements. Each comment indicates the time of day and sequence number of the record in the IDC log data set used to generate the statements following the comment.

A sample comment for STL programs is shown below.

```
/*-------------------------------------------------------- 07582389 00001 */
```

A sample comment for WSim message generation decks is shown below.

```
*------------------------------------------------------------- 07582389 00001
```

In the above example, the first number is the time of day, in the format HHMMSSTH. The second number is the sequence number of the log record used to generate the statements following the comment.

Use the DEBUG parameter as a debugging aid for script generation. If you do not specify the DEBUG parameter, these comments are not inserted into the generated scripts.

NOWTO
Suppresses WTO (Write to Operator) informational messages while IDC is executing. You can use this execution parameter if you submit the IDC job stream as a batch job and do not want the IDC messages to appear on the system console.

NOWTOR
Suppresses WTORs (Write to Operator with Reply) while IDC is executing. Coding NOWTOR allows you to run ITPIDC without operator intervention. When IDC runs under MVS as a submitted batch job or under TSO without the TSOCON option, IDC issues a WTOR to allow you to respond with the END command when it is time to end IDC. When running under MVS as a started procedure, IDC does not issue a WTOR and you can use an MVS stop (P) or modify (F) END command to end IDC. Code NOWTOR when you do not want operator intervention and plan to end IDC by selecting option 4 from the IDC Main panel (see Figure 37 on page 182). You can always end IDC by selecting option 4 from the IDC Main panel.

PRTLNCNT=*nnn*
Specifies the maximum number of lines that can be printed on a page of SYSPRINT output before ejecting to a new page. *nnn* is an integer from 35 to 255. The default value for *nnn* is 60.

ROUTCDE=(*n*,*n*,...)
Specifies the system message routing codes to be used when IDC writes messages to the operator. Each *n* is a system routing code that defines a console destination for every WTO and WTOR message that IDC writes. *n* is an integer from 1 to 16. The default value for ROUTCDE is 8.

TRACE
Creates a log of all traffic on both IDC sessions (PLU and SLU) in the IDC trace data set. You must code an IDCTRACE DD statement in the IDC job stream to use this option.

This parameter is usually used for debug purposes only. If you do not specify TRACE, IDC does not write any data to the trace data set.

TSOCON
Allows you to run ITPIDC from the TSO console. When you specify TSOCON, you do not need to log on to ITPIDC; the IDC Main panel is immediately displayed (see Figure 37 on page 182). If you want to logon to ITPIDC as a VTAM application, do not specify TSOCON. Also, if you specify TSOCON when TSO is not active, it is ignored.

Note: You must have TSO/VTAM on your system to use TSOCON.

Establishing sessions

Once IDC is running, you may log on to IDC and then, through IDC, log on to the host application.

Logging on to IDC

Log on to IDC using the procedures defined at your installation for VTAM applications (IDC does not provide any built-in security, such as passwords or authorizations to restrict user access.)

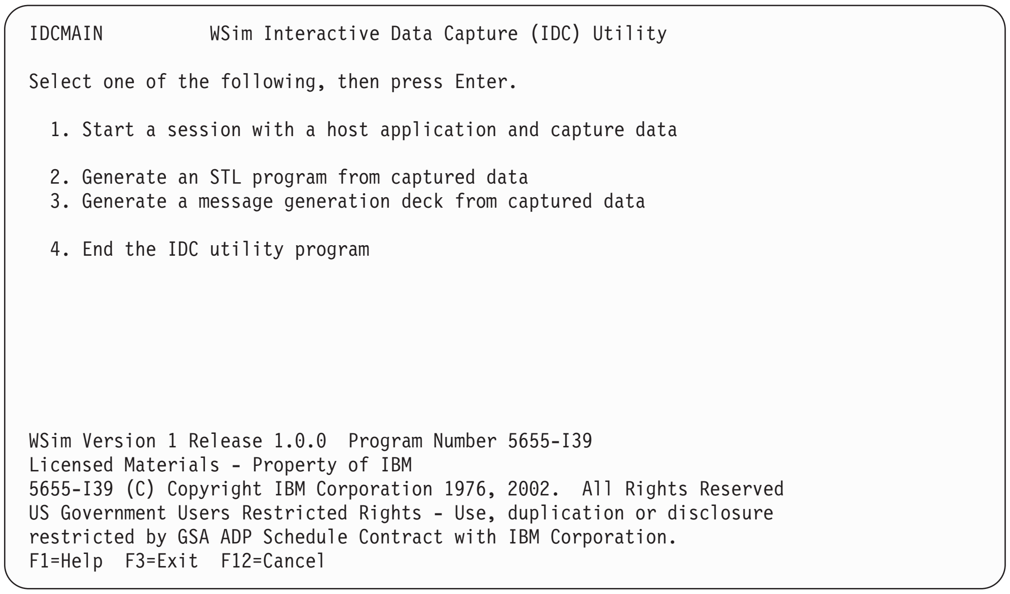
When logon processing completes, IDC displays the panel shown in Figure 37 on page 182. To select an option on this panel, either type the option number in the entry field or use the tab or new line keys to position the cursor on the desired option. Press **Enter** to complete the selection.

From this panel, you can select the following options:

1. **Start a session with a host application and capture data.**
   This option displays the IDC Start Session panel (Figure 38 on page 183).
2. **Generate an STL program from captured data.**
   This option displays the IDC Generate STL Program panel (Figure 43 on page 191).
3. **Generate a message generation deck from captured data.**
   This option displays the IDC Generate Message Generation Deck panel (Figure 44 on page 195).
4. **End the IDC utility program.**
   This option ends the IDC program (and, of course, ends your session with IDC). You can use this option when you want to end the IDC program without intervention from the system operator. Do not confuse this with the **F3** or **F12** actions, which only end your session with IDC.

You can use the following function keys on this panel:

F1 (Help)	Display help information for this panel (see "Using help" on page 206).
F3 (Exit)	End your session with IDC. This logs you off IDC, but the program itself remains active and ready to accept another logon.
F12 (Cancel)	For this panel, **F12** invokes the same action as **F3** (Exit).



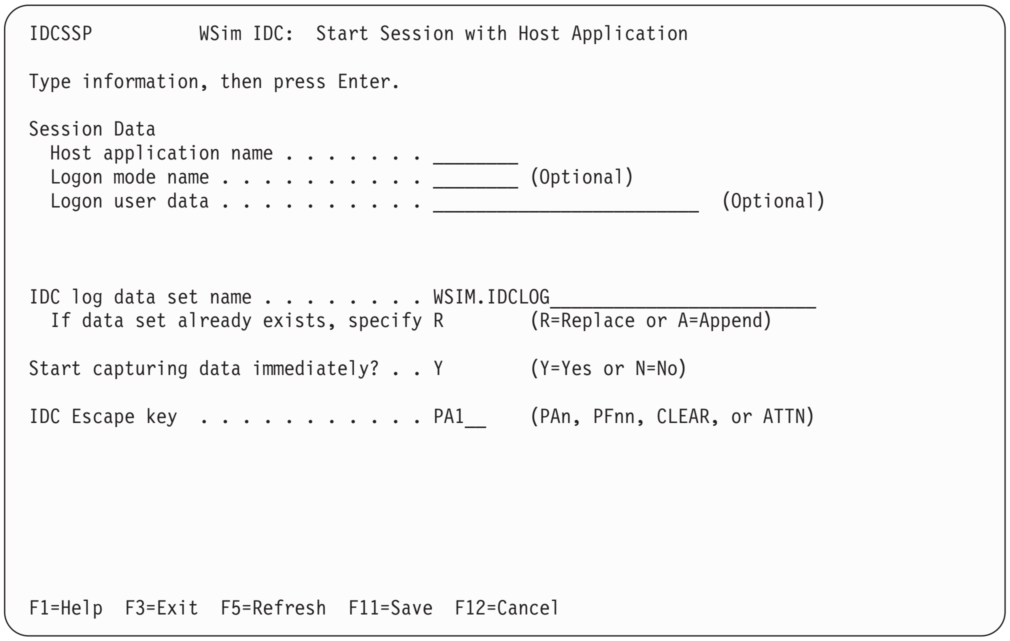
```
IDCMAIN          WSim Interactive Data Capture (IDC) Utility

Select one of the following, then press Enter.

  1. Start a session with a host application and capture data

  2. Generate an STL program from captured data
  3. Generate a message generation deck from captured data

  4. End the IDC utility program








WSim Version 1 Release 1.0.0  Program Number 5655-I39
Licensed Materials - Property of IBM
5655-I39 (C) Copyright IBM Corporation 1976, 2002.  All Rights Reserved
US Government Users Restricted Rights - Use, duplication or disclosure
restricted by GSA ADP Schedule Contract with IBM Corporation.
F1=Help  F3=Exit  F12=Cancel
```

Figure 37. IDC main panel

Note: If you specified the TSOCON execution parameter, you do not need to logon to IDC. The IDC Main panel is immediately displayed.

Logging on to your application

When you select option 1 from the IDC Main panel (Figure 37), IDC displays the IDC Start Session panel (Figure 38 on page 183). The IDC Start Session panel establishes the session between IDC and the host application.



```
IDCSSP         WSim IDC:  Start Session with Host Application

Type information, then press Enter.

Session Data
  Host application name . . . . . . . ________
  Logon mode name . . . . . . . . . . ________ (Optional)
  Logon user data . . . . . . . . . . _________________________  (Optional)



IDC log data set name . . . . . . . . WSIM.IDCLOG_________________________
  If data set already exists, specify R        (R=Replace or A=Append)

Start capturing data immediately? . . Y        (Y=Yes or N=No)

IDC Escape key  . . . . . . . . . . . PA1__    (PAn, PFnn, CLEAR, or ATTN)






F1=Help  F3=Exit  F5=Refresh  F11=Save  F12=Cancel
```

Figure 38. IDC start session panel

Fill in the following entry fields and press **Enter** when you finish.

`Host application name`
: Type the name of the host application to which you want to logon and capture data.

`Logon mode entry name`
: Type the name of the logon mode table entry to be used when establishing the host application session. This field is optional; the default is the logon mode name used when you logged on to IDC. If you use the TSOCON option, the logon mode that best fits the characteristics of the TSO console is displayed. One of the following logon modes is displayed: LSX32702, LSX32703, LSX32704, LSX32705, or D329001. These logon modes are in the standard VTAM logon mode table. Unless you choose to override the logon mode name, the logon mode table you use should contain each of these modes defined as they are in the standard VTAM logon mode table. If you use a different logon mode name, be sure to use a logon mode with compatible session protocols.

`Logon user data`
: Type any other data you want to enter in the host application logon sequence, such as a user ID. This field is optional; the default is the user data you entered (if any) when you logged onto IDC, up to 25 bytes.

`Log data set name`
: Type the name of the data set where you will save the captured data.

: Type a name up to 36 characters in length, including a member name in parentheses if you are using a partitioned data set. Do not enclose the name in quotes. You must allocate and catalog this data set in advance (see “Setting up IDC” on page 176).

`Append captured data to IDC log data set`
: Type an “R” (replace) to write over any previously captured data in the log data set. Type an “A” to append new captured data to the log data set. (If you use a partitioned log data set, you can only specify replace. Replace will create a new member if the member does not already exist.)

`Start capturing data when session active`
Type "Y" to automatically start capturing data when the host application session first becomes active. Type "N" to manually control when IDC starts capturing data. If you type "N", you can manually start data capture from the IDC Escape Actions panel (see "Using escape actions").

`IDC Escape key`
Type the name of the key you want to use to toggle between IDC and the host application. You can enter any PA key, any function key, CLEAR, or ATTN in this field. Choose a key that is available on your keyboard, but not used by your application. If that is not possible, the IDC escape key can be passed to your host application when necessary (see "Using escape actions").

Note: When using ATTN as the IDC escape key, PROG416, PROG707, or PROG709 may briefly appear in the operator information area of your display. Also, you cannot use ATTN as a valid escape key when you run IDC with the TSOCON option.

You can use the following function keys on this panel:

F1 (Help)	Display help information for this panel (see "Using help" on page 206).
F3 (Exit)	End your session with IDC. This logs you off IDC, but the program itself remains active and ready to accept another logon.
F5 (Refresh)	Restore previously saved values to the entry fields and erase all other entries.
F11 (Save)	Save the current values of the entry fields.
F12 (Cancel)	Return to the IDC Main panel.

When IDC establishes the host session, data from the host application appears on the display and IDC becomes transparent to both you and your host application. [1]

If the host session cannot be started successfully, one or more error messages appear in the message area at the bottom of the panel. Numbered messages are described in *WSim Messages and Codes*.

Capturing the data

Once you log on to a host application through IDC, you can start capturing session data. The sections below discuss how to do this.

Capturing the session initiation

If you request to immediately capture data on the IDC Start Session panel, IDC captures the session initiation SNA data flows between the host application and IDC. IDC uses this data to generate the logon sequence in the WSim script.

Using escape actions

If you need access to IDC while you are in session with the host application, press the escape key you defined on the IDC Start Session panel (Figure 38 on page 183). IDC responds to the escape key by displaying the IDC Escape Actions panel (Figure 39 on page 185).

1. IDC is not always completely transparent. See "Understanding IDC restrictions" on page 207.

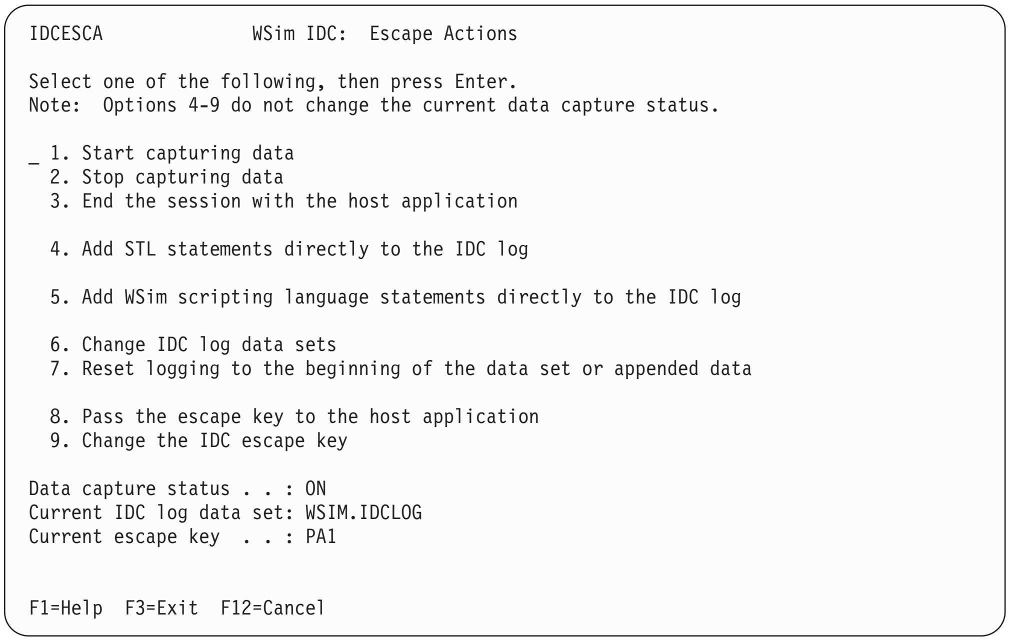

```
IDCESCA              WSim IDC:  Escape Actions

Select one of the following, then press Enter.
Note:  Options 4-9 do not change the current data capture status.

_ 1. Start capturing data
  2. Stop capturing data
  3. End the session with the host application

  4. Add STL statements directly to the IDC log

  5. Add WSim scripting language statements directly to the IDC log

  6. Change IDC log data sets
  7. Reset logging to the beginning of the data set or appended data

  8. Pass the escape key to the host application
  9. Change the IDC escape key

Data capture status . . : ON
Current IDC log data set: WSIM.IDCLOG
Current escape key  . . : PA1


F1=Help  F3=Exit  F12=Cancel
```

Figure 39. IDC escape actions panel

To select an option on this panel, either type the option number in the entry field or use the tab or new line key to position the cursor on the desired option. Press **Enter** to complete the selection.

You can select the following options from this panel:

1. **Start capturing data.**
2. **Stop capturing data.**
3. **End the session with the host application.**

   This option terminates the session between IDC and the host application and displays the IDC Start Session panel (Figure 38 on page 183).
4. **Add STL statements directly to the IDC log.**

   This option displays the IDC Add STL Statements panel (Figure 40 on page 187).
5. **Add WSim scripting language statements directly to the IDC log.**

   This option displays the IDC Add Scripting Language Statements panel (Figure 40 on page 187).
6. **Change IDC log data sets.**

   This option displays the IDC Change Log Data Sets panel (Figure 41 on page 188).
7. **Reset logging to the beginning of the data set or appended data.**

   This option allows you to start over if you make a mistake while capturing data. If you are writing to a new IDC log data set or replacing an existing IDC log data set, logging starts over at the beginning of the data set. If you are appending data to an existing IDC log data set, logging starts over at the beginning of the appended data.
8. **Pass the escape key to the host application.**

   This option sends the data generated by the escape key to the host application. If you press the escape key during an IDC session, IDC intercepts the escape key and does not pass it to the host application.
9. **Change the IDC escape key.**

This option displays the IDC Change Escape Key panel (Figure 42 on page 190).

You can use the following function keys on this panel:

F1 (Help)	Display help information for this panel (see “Using help” on page 206).
F3 (Exit)	Leave this panel and return to the host application session.
F12 (Cancel)	For this panel, **F12** invokes the same action as **F3** (Exit).

Controlling data capture

You may not want to capture all of your interactions with the host application. Rather, you may want to create a script that only performs a specific transaction, such as querying an account balance in a banking application. To do this, do not begin capturing data immediately. Instead, wait to start capturing data until you are logged on and ready to invoke the query transaction. You then start capturing data, invoke the query transaction, and when the transaction completes, stop capturing data.

The sequence of actions you perform is described below:

1. Specify “N” (No) for “Start capturing data immediately?” on the IDC Start Session panel (Figure 38 on page 183).
2. Interact with the host application to get to the panel that performs the query operation.
3. Press the IDC escape key. IDC displays the IDC Escape Actions panel (Figure 39 on page 185). Data capture status is “OFF” at this point.
4. Select option 1 to start capturing data. Verify that the capture status changes to “ON”.
5. Press **F12** to return to the host application query panel.
6. Perform the query transaction (or transactions).
7. Press the IDC escape key.
8. Select option 2 to stop capturing data.
9. Press **F12** to return to the host application query panel.
10. Log off the application normally.

Adding statements to the IDC log

IDC allows you to add statements directly to the IDC log. When you generate the script, IDC copies these statements to the script at the same point in the data as they were added to the IDC log.

You can add either STL statements or WSim Scripting Language statements. There is a separate IDC Escape Actions panel option and corresponding entry panel for each. If you add STL statements to the IDC log and then decide to generate a message generation deck instead of an STL program, the STL statements are ignored. The same is true for WSim Scripting Language statements if you generate an STL program instead of a message generation deck.

Some examples of statements (in STL) that you may add are shown below:

- Comments

  ```
  /* Test script number 1, created by Jane Smith on 7/1/02. */
  ```

- Messages written to the operator's console

  ```
  say 'End of query testing, starting update testing.'
  ```

- WSim operator commands

```
opcmnd('ZEND')  /* Close down WSim */
```

- Logic tests. The following STL statements will wait for the specified data to be displayed on the screen.

```
check_data = 'ISPF/PDF PRIMARY OPTION MENU'
do while index(screen,check_data) = 0
   wait until onin
end
```

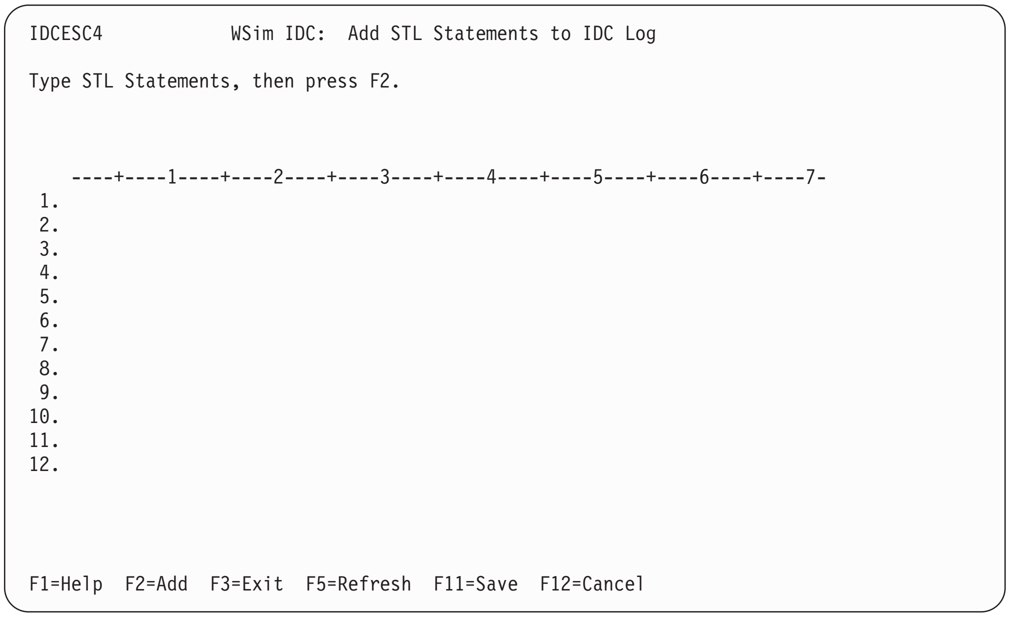
Note: IDC does not perform syntax checking on the statements you add! IDC adds your statements exactly as you enter them.

If you make an error in an added STL statement, it will be flagged when you run the generated STL program through the STL Translator. If you make an error in an added WSim Scripting Language statement, it will be flagged when you preprocess the network and message decks with the Preprocessor Utility. If you generate WSim message generation decks and have IDC place them directly in the WSim MSGDD data set, any errors will be flagged when the network referencing the decks is initialized. In all these cases, you can check the error message and correct the problem in the generated script without having to recapture any of the data.

Selecting option 4 or 5 from the IDC Escape Actions panel (Figure 39 on page 185) displays an IDC Add Statements panel. The panel for STL is shown in Figure 40; there is a similar panel for the WSim Scripting Language.

If you plan to add numerous statements to the script, generate the script first and then add the statements later using an editor.

For information about the WSim Scripting Language and STL, see *Creating WSim Scripts* and *WSim Script Guide and Reference*.



```
IDCESC4           WSim IDC:  Add STL Statements to IDC Log

Type STL Statements, then press F2.



   ----+----1----+----2----+----3----+----4----+----5----+----6----+----7-
 1.
 2.
 3.
 4.
 5.
 6.
 7.
 8.
 9.
10.
11.
12.




F1=Help  F2=Add  F3=Exit  F5=Refresh  F11=Save  F12=Cancel
```

Figure 40. IDC add STL statements panel

You can use the following function keys on this panel:

F1 (Help)	Display help information for this panel (see "Using help" on page 206).
F2 (Add)	Add the statements to the IDC log.

F3 (Exit)	End escape actions and return to the host application.
F5 (Refresh)	Restore previously saved values to the entry fields.
F11 (Save)	Save the current values of the entry fields.
F12 (Cancel)	Return to the IDC Escape Actions panel.

Note: Pressing **F3** or **F12** before pressing **F2** causes any statements you typed to be discarded.

Changing log data sets

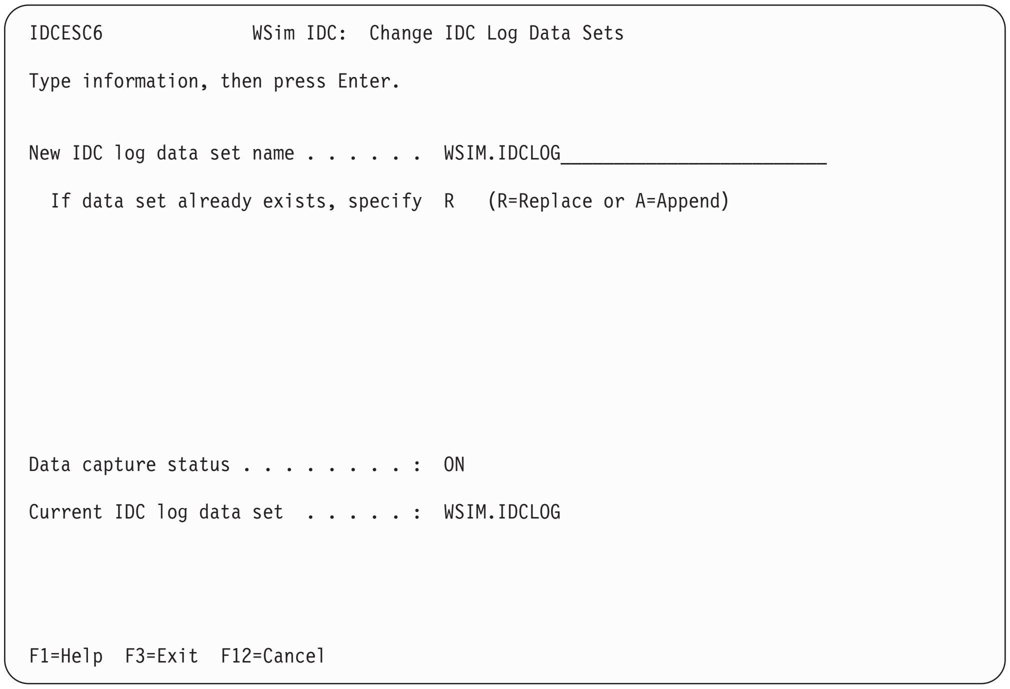
One IDC log can only create one script. If you want to create multiple scripts from a single capture session, you must create multiple IDC logs.

For example, you may want to create separate scripts for entering, updating, and querying data in a database. To do this, you can log on to the database application, capture the data for the entry operation, log off the application, generate the script, and then repeat those steps for the update and the query operations.

Alternately, you can log on to the database application, capture each operation in a separate IDC log data set or member, log off the application, and then generate the scripts. This eliminates the need to log on and off the application for each script.

Selecting option 6 from the IDC Escape Actions panel (Figure 39 on page 185) displays the IDC Change Log Data Sets panel (Figure 41).

Note: You must allocate and catalog data sets in advance; IDC will not do it for you. However, if you use a partitioned data set, you can add new members to or replace existing members in an existing data set.
Fill in the following fields and press **Enter**:



```
IDCESC6             WSim IDC:  Change IDC Log Data Sets

Type information, then press Enter.


New IDC log data set name . . . . . .  WSIM.IDCLOG________________________

  If data set already exists, specify  R   (R=Replace or A=Append)











Data capture status . . . . . . . . :  ON

Current IDC log data set  . . . . . :  WSIM.IDCLOG





F1=Help  F3=Exit  F12=Cancel
```

Figure 41. IDC change log data sets panel

`New IDC log data set name`
Type the name of the data set where you will save the captured data.

Type a name up to 36 characters in length, including a member name in parentheses if you are using a partitioned data set. Do not enclose the name in quotes. You must allocate and catalog this data set in advance (see “Setting up IDC” on page 176).

`Replace or Append captured data to IDC log data set`
Type an “R” (replace) to write over any previously captured data in the log data set. Type an “A” to append new captured data to the log data set. (If you use a partitioned log data set, you can only specify replace.)

You can use the following function keys on this panel:

F1 (Help)	Display help information for this panel (see “Using help” on page 206).
F3 (Exit)	End escape actions and return to the host application.
F12 (Cancel)	Return to the IDC Escape Actions panel.

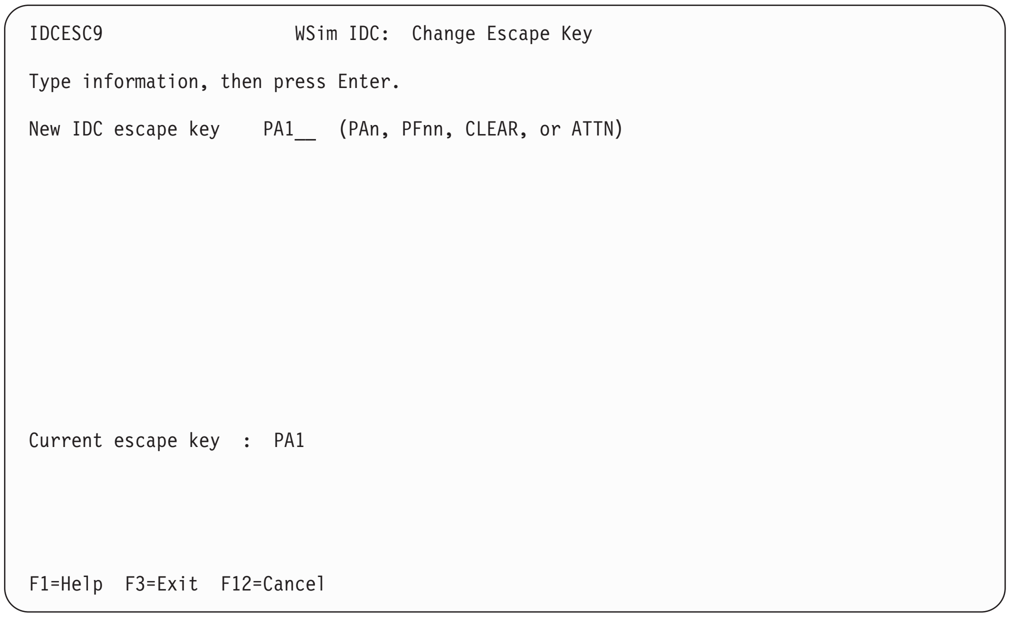
Note: Pressing **F3** or **F12** before pressing **Enter** cancels the change.

Changing escape keys

You may discover after you log on to your host application that you need to change the IDC escape key. For example, you assign F5 as the IDC escape key and find that you use F5 repeatedly for the host application. To change your escape key, select option 9 from the IDC Escape Actions panel (Figure 39 on page 185). This displays the IDC Change Escape Key panel (Figure 42 on page 190). Be sure to choose a key that is actually available on your keyboard. IDC does not object if you choose F13 when your keyboard only has 12 function keys. You cannot use ATTN as a valid escape key when you run IDC with the TSOCON option.

(You can also use option 8 from the IDC Escape Actions panel to send the escape key to the host application.)

Note: If you change the escape key and then immediately select option 8 on the IDC Escape Actions panel to send the new escape key to the host application, you will get an error message. This is because IDC needs to receive the actual data the display would send for the new escape key.



```
IDCESC9                 WSim IDC:  Change Escape Key

Type information, then press Enter.

New IDC escape key    PA1__  (PAn, PFnn, CLEAR, or ATTN)















Current escape key  :  PA1






F1=Help  F3=Exit  F12=Cancel
```

Figure 42. IDC change escape key panel

You can use the following function keys on this panel:

F1 (Help)	Display help information for this panel (see "Using help" on page 206).
F3 (Exit)	End escape actions and return to the host application.
F12 (Cancel)	Return to the IDC Escape Actions panel.

Note: Pressing **F3** or **F12** before pressing **Enter** cancels the change.

Capturing session termination

There are two ways to capture a session termination sequence.

1. Log off the application normally. IDC converts all of the user actions into statements in the generated script.
2. Select option 3 from the IDC Escape Actions panel (Figure 39 on page 185). The program converts the captured SNA session termination flows into the WSim TERMSESS command in the generated script.

Stopping IDC

You can use the following methods to stop IDC:

1. Reply to the IDC WTOR at the operator's console with the "END" command.
2. Select option 4 from the IDC Main panel (Figure 37 on page 182).
3. Issue the MVS stop (P) command or the MVS modify (F) command with the "END" operand when IDC is running on MVS as a started procedure.

Note: If you specified the TSOCON execution parameter, you can select option 4, press F3, or press F12 from the IDC Main panel to stop IDC.

Generating scripts

When you finish capturing your data, you can generate either a STL program or a WSim Scripting Language message generation deck. The following sections describe how to generate a script.

Choosing the type of script to generate

As mentioned earlier, IDC creates either an STL program or a message generation deck. The choice of format depends on how you plan to use the scripts.

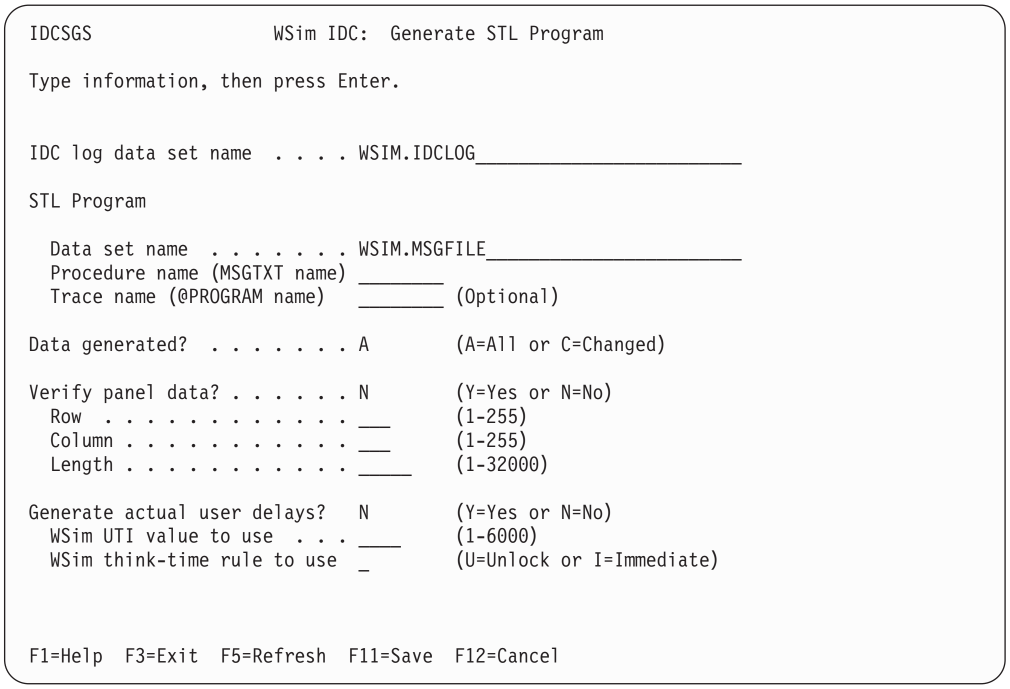
Generate an STL program if you expect to modify the script created by IDC. If you are a new user, you should write scripts in STL since it is easier to learn and use. You must process the STL program created by IDC using the STL Translator. The STL Translator converts the STL program into a message generation deck which can be executed by WSim.

Generate a message generation deck if you do not plan to modify the scripts created by IDC or if you are already familiar with the WSim Scripting Language. Since the message generation deck is directly executable by WSim, you eliminate the translation step required when using STL.

You can read about the WSim Scripting Language and STL in *Creating WSim Scripts* and *WSim Script Guide and Reference*.

Generating an STL program

Selecting option 2 from the IDC Main panel (Figure 37 on page 182) displays the IDC Generate STL Program panel (Figure 43).



```
IDCSGS                WSim IDC:  Generate STL Program

Type information, then press Enter.


IDC log data set name  . . . . WSIM.IDCLOG_________________________

STL Program

  Data set name  . . . . . . . WSIM.MSGFILE________________________
  Procedure name (MSGTXT name) ________
  Trace name (@PROGRAM name)   ________ (Optional)

Data generated?  . . . . . . . A        (A=All or C=Changed)

Verify panel data? . . . . . . N        (Y=Yes or N=No)
  Row  . . . . . . . . . . . . ___      (1-255)
  Column . . . . . . . . . . . ___      (1-255)
  Length . . . . . . . . . . . _____    (1-32000)

Generate actual user delays?   N        (Y=Yes or N=No)
  WSim UTI value to use  . . . ____     (1-6000)
  WSim think-time rule to use  _        (U=Unlock or I=Immediate)



F1=Help  F3=Exit  F5=Refresh  F11=Save  F12=Cancel
```

Figure 43. IDC generate STL program panel

Fill in the following fields and press **Enter**.

`IDC log data set name`
: Type the name of the data set containing the captured data.

Type a name up to 36 characters in length, including a member name in parentheses if you are using a partitioned data set. Do not enclose the name in quotes.

STL program data set name
Type the name of the data set in which you want the generated STL program to be placed.

Type a name up to 36 characters in length, optionally including a member name in parentheses if you are using a partitioned data set. Do not enclose the name in quotes. You must allocate and catalog this data set in advance (see "Setting up IDC" on page 176).

STL program procedure name (MSGTXT name)
Type the procedure name of the STL program. This name appears as the label for the MSGTXT statement. This is also the default name of the member created if you are using a partitioned data set for the STL programs.

The name must be 1 to 8 alphanumeric or special ($,@,_,?,#) characters, where the first character is non-numeric. The name cannot be an STL reserved word or begin with $INC, $LA, or $SET (reserved labels in STL). Also, you cannot use the same name as the STL program trace name.

STL program trace name (@PROGRAM name)
Type the name you want to appear on the STL @PROGRAM statement. (The @PROGRAM statement defines a symbolic name used to label STL trace records.) This field is required if you want to trace the execution of your STL program at WSim run time or query the STL statement number during execution.

The name must be 1 to 8 alphanumeric or special ($,@,_,?,#) characters, where the first character is non-numeric. The name cannot be an STL reserved word or begin with $INC, $LA, or $SET (reserved labels in STL). Also, you cannot use the same name as the STL program procedure name.

Data generated?
Type an "A" if you want IDC to generate scripting statements for all the unprotected fields on the screen containing data sent to the application program. This includes unprotected fields set by the application with the MDT bit on that you did not change during the data capture session. Type a "C" if you want IDC to generate scripting statements for only the unprotected fields that you actually changed during the data capture session. Unprotected fields set by the application with the MDT bit on are ignored for script generation.

Note: Unprotected fields with the 3270 Modified Data Tag (MDT) bit set on are sent to the application program when you press Enter or a function key.

Verify panel data?
Type a "Y" if you want IDC to generate a script with statements that test data in the specified panel location (row and column). Type an "N" if you do not want IDC to generate panel verification statements. See "Adding panel verification logic to the script" on page 197 for a discussion of this option.

Row
Type the row number of the panel data to be verified. This must be a number between 1 and 255.

Column
Type the column number of the panel data to be verified. This must be a number between 1 and 255.

Length
: Type the length of the data, starting at the row and column numbers stated above, that you want verified. The maximum is 32000 characters.

Generate actual user delays?
: Type a “Y” if you want IDC to generate DELAY statements to reproduce the recorded user delays. Type an “N” if you do not want DELAY statements to be generated. See “Generating user delays” on page 202 for a discussion of this option and *Creating WSim Scripts* for a discussion of WSim intermessage delays.

WSim UTI value to use
: Type the user time interval value (in hundredths of a second) to use in calculating delays. This must be a number between 1 (representing .01 seconds) and 6000 (representing 1 minute).

WSim think-time rule to use
: Type an “I” if you want IDC to use the immediate think-time rule for calculating delays. Type a “U” if you want IDC to use the unlock think-time rule for calculating delays.

  Note: If you want to simulate LU type 2 devices, type a “U” here and code THKTIME=UNLOCK in your network definition.

Remember, you must translate STL programs (using the STL Translator Utility) before they can be executed by WSim.

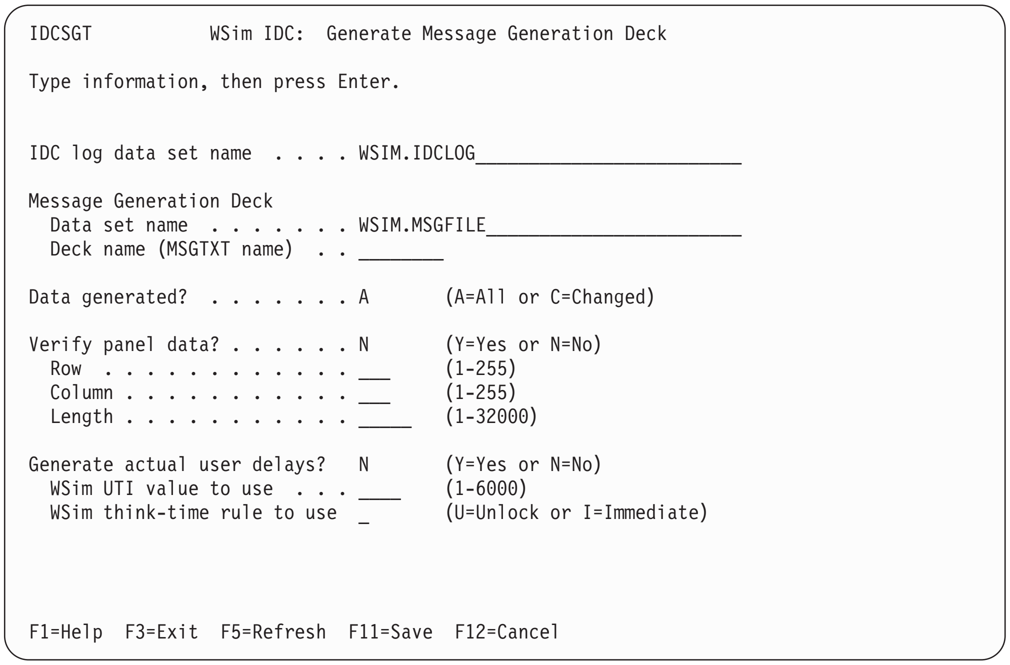
Below is an example of an STL program created by IDC.

```
@program=STLDECKT
STLDECK: msgtxt
/*-----------------------------------------------------------------*/
/* ITPIDC: DISPLAY=WSIM420  APPLICATION=TSO01    07:58:23.89 02/16/02*/
/*   ----------- DISPLAY CHARACTERISTICS AND FEATURES ------------   */
/*   ALTCSET=APL                APLCSID=(963,310)                   */
/*   BASECSID=(697,37)          CCSIZE=(12,22)    COLOR=MULTI       */
/*   DBCS=NO                                                        */
/*   DISPLAY=(24,80,24,80)      DLOGMOD=NSX32702  EXTFUN=YES        */
/*   FLDOUTLN=NO                FLDVALID=NO       HIGHLITE=YES      */
/*   MAXNOPTN=0                 PS=NONE           UOM=INCH          */
/*-----------------------------------------------------------------*/
/* ITPLSGEN: SCRIPT GENERATION PARAMETERS        08:05:35.00 02/16/02*/
/*   INPUT    WSIM.IDCLOG                                           */
/*   OUTPUT   WSIM.STLFILE                                          */
/*   MSGTXT   STLDECK                                               */
/*   NODELAY                                                        */
/*   GENERATE ALL                                                   */
/*   LU       IDCSLU-1                                              */
/*   STL      TRACE=STLDECKT                                        */
/*   NOVERIFY                                                       */
/*-----------------------------------------------------------------*/
onin0001: onin substr(ru,1,1) = 'F5'x,
               then found = on
found = off
initself('TSO01','NSX32702')
do while found = off             /* wait for onin0001 data received */
 wait until onin
end
deact onin0001
```

```
/* 07:58:29.36 ITP1507I SESSION STARTED WITH APPLICATION TSO01 */

/* 07:58:31.83 ITP1508I SESSION ENDED WITH APPLICATION TSO01 */

/* 07:58:31.99 ITP1507I SESSION STARTED WITH APPLICATION TSO0102 */
cursor(1,27)
ereof
cursor(2,1)
type 'user5'
transmit using enter
transmit using enter
transmit using clear
cursor(2,15)
ereof
type '3.4'
transmit using enter
cursor(8,23)
ereof
type 'sys1'
transmit using enter
cursor(2,15)
ereof
type '=x'
transmit using enter
cursor(1,9)
ereof
cursor(2,1)
type 'logoff'
transmit using enter
/* 08:00:25.30 ITP1508I SESSION ENDED WITH APPLICATION TSO0102 */

endtxt
```

Generating a message generation deck

Selecting option 3 from the IDC Main panel (Figure 37 on page 182) displays the IDC Generate Message Generation Deck panel (Figure 44 on page 195).



```
IDCSGT          WSim IDC:  Generate Message Generation Deck

Type information, then press Enter.


IDC log data set name  . . . . WSIM.IDCLOG_________________________

Message Generation Deck
  Data set name  . . . . . . . WSIM.MSGFILE________________________
  Deck name (MSGTXT name)  . . ________

Data generated?  . . . . . . . A       (A=All or C=Changed)

Verify panel data? . . . . . . N       (Y=Yes or N=No)
  Row  . . . . . . . . . . . . ___     (1-255)
  Column . . . . . . . . . . . ___     (1-255)
  Length . . . . . . . . . . . _____   (1-32000)

Generate actual user delays?   N       (Y=Yes or N=No)
  WSim UTI value to use  . . . ____    (1-6000)
  WSim think-time rule to use  _       (U=Unlock or I=Immediate)




F1=Help  F3=Exit  F5=Refresh  F11=Save  F12=Cancel
```

Figure 44. IDC generate message generation deck panel

Fill in the following fields and press **Enter**:

`IDC log data set name`
: Type the name of the data set containing the captured data.

  Type a name up to 36 characters in length, including a member name in parentheses if you are using a partitioned data set. Do not enclose the name in quotes.

`Message Generation Deck data set name`
: Type the name of the data set in which you want the message generation deck to be placed.

  Type a name up to 36 characters in length, optionally including a member name in parentheses if you are using a partitioned data set. Do not enclose the name in quotes. You must allocate and catalog this data set in advance (see "Setting up IDC" on page 176).

`Message Generation Deck name (MSGTXT name)`
: Type the name of the message generation deck. This name appears in the name field of the MSGTXT statement. This is also the default name of the member created if you are using a partitioned data set for the message generation deck.

  The name must be 1 to 8 alphanumeric or special ($,@,_,?,#) characters, where the first character is non-numeric.

`Data generated?`
: Type an "A" if you want IDC to generate scripting statements for all the unprotected fields on the screen containing data sent to the application program. This includes unprotected fields set by the application with the MDT bit on that you did not change during the data capture session. Type a "C" if you want IDC to generate scripting statements for only the unprotected fields that you actually changed during the data capture session. Unprotected fields set by the application with the MDT bit on are ignored for script generation.

  Note: Unprotected fields with the 3270 Modified Data Tag (MDT) bit set on are sent to the application program when you press Enter or a function key.

Verify panel data?
Type a "Y" if you want IDC to generate a script with statements to test data in the specified panel location (row and column). Type an "N" if you do not want IDC to generate panel verification statements. (See "Adding panel verification logic to the script" on page 197 for a discussion of this option.)

Row
Type the row number of the panel data to be verified. This must be a number between 1 and 255.

Column
Type the column number of the panel data to be verified. This must be a number between 1 and 255.

Length
Type the length of the data, starting at the row and column numbers stated above, that you want verified. The maximum is 32000 characters.

Generate actual user delays?
Type a "Y" if you want IDC to generate DELAY statements to reproduce the recorded user delays. Type an "N" if you do not want DELAY statements to be generated. (See "Generating user delays" on page 202 for a discussion of this option and *Creating WSim Scripts* for a discussion of WSim intermessage delays.)

WSim UTI value to use
Type the user time interval value (in hundredths of a second) to use in calculating delays. This must be a number between 1 (representing .01 seconds) and 6000 (representing 1 minute).

WSim think-time rule to use
Type an "I" if you want IDC to use the immediate think-time rule for calculating delays. Type a "U" if you want IDC to use the unlock think-time rule for calculating delays.

Remember, message generation decks are written in the WSim Scripting Language, which is directly executable by WSim.

Below is an example of a message generation deck created by IDC.

```
WSIMDECK MSGTXT
*---------------------------------------------------------------------
* ITPIDC: DISPLAY=WSIM420  APPLICATION=TSO01    07:58:23.89 02/16/02
*   ----------- DISPLAY CHARACTERISTICS AND FEATURES ------------
*   ALTCSET=APL               APLCSID=(963,310)
*   BASECSID=(697,37)         CCSIZE=(12,22)    COLOR=MULTI
*   DBCS=NO
*   DISPLAY=(24,80,24,80)     DLOGMOD=NSX32702  EXTFUN=YES
*   FLDOUTLN=NO               FLDVALID=NO       HIGHLITE=YES
*   MAXNOPTN=0                PS=NONE           UOM=INCH
*---------------------------------------------------------------------
* ITPLSGEN: SCRIPT GENERATION PARAMETERS       08:05:35.00 02/16/02
*   INPUT    WSIM.IDCLOG
*   OUTPUT   WSIM.MSGFILE
*   MSGTXT   WSIMDECK
*   NODELAY
*   GENERATE ALL
*   LU       IDCSLU-1
*   WSIM
*   NOVERIFY
*---------------------------------------------------------------------


         CMND     COMMAND=INIT,RESOURCE=TSO01,MODE=NSX32702
```

```
0        IF       LOC=RU+0,TEXT=('F5'),
                  THEN=B-CONT0001
WAIT0001 WAIT
         BRANCH   LABEL=WAIT0001
CONT0001 DEACT    IFS=(0)


* 07:58:29.36 ITP1507I SESSION STARTED WITH APPLICATION TSO01

* 07:58:31.83 ITP1508I SESSION ENDED WITH APPLICATION TSO01

* 07:58:31.99 ITP1507I SESSION STARTED WITH APPLICATION TSO0102


         CURSOR   ROW=1,COLUMN=27
         EREOF
         CURSOR   ROW=2,COLUMN=1
         TEXT     (user5)
         ENTER


         ENTER


         CLEAR


         CURSOR   ROW=2,COLUMN=15
         EREOF
         TEXT     (3.4)
         ENTER

         CURSOR   ROW=8,COLUMN=23
         EREOF
         TEXT     (sys1)
         ENTER


         CURSOR   ROW=2,COLUMN=15
         EREOF
         TEXT     (=x)
         ENTER


         CURSOR   ROW=1,COLUMN=9
         EREOF
         CURSOR   ROW=2,COLUMN=1
         TEXT     (logoff)
         ENTER


* 08:00:25.30 ITP1508I SESSION ENDED WITH APPLICATION TSO0102

         ENDTXT
```

Adding panel verification logic to the script

You may choose to have IDC generate statements to test data appearing at a specific location on all panels. This allows you to verify that the panels encountered during data capture are the same panels encountered when the script is actually run.

You can select this option from either the IDC Generate STL Program panel (Figure 43 on page 191) or the IDC Generate Message Generation Deck panel (Figure 44 on page 195).

You can only specify a single location and data length for all panels. For this reason, you may want to specify a location that contains predictable information, such as a panel identifier.

The amount of data verified can cross rows. For example, to verify the second and third rows on an 80-column screen, specify a starting location of row 2, column 1, with a length of 160. If you specify a length that exceeds the screen size, the length value is reduced to match that size.

If the test fails during execution, WSim writes a VERIFY record to the WSim log and stops execution of the script. When the Loglist Utility processes the WSim log, it formats the VERIFY record and prints it out with all of the other records in the log. (For a description of the Loglist Utility and all of the records in the WSim log, see Part 1, "General utilities," on page 1.). The VERIFY record indicates the number of the panel tested, the location and length of the test, and the expected and actual values of the data.

Below is an example of an STL program with panel verification.

```
@program=STLDECKT
STLDECK: msgtxt
/*------------------------------------------------------------------*/
/* ITPIDC: DISPLAY=WSIM420  APPLICATION=TSO01    07:58:23.89 02/16/02*/
/*   ----------- DISPLAY CHARACTERISTICS AND FEATURES ------------   */
/*   ALTCSET=APL               APLCSID=(963,310)                     */
/*   BASECSID=(697,37)         CCSIZE=(12,22)    COLOR=MULTI         */
/*   DBCS=NO                                                         */
/*   DISPLAY=(24,80,24,80)     DLOGMOD=NSX32702  EXTFUN=YES          */
/*   FLDOUTLN=NO               FLDVALID=NO       HIGHLITE=YES        */
/*   MAXNOPTN=0                PS=NONE           UOM=INCH            */
/*------------------------------------------------------------------*/
/* ITPLSGEN: SCRIPT GENERATION PARAMETERS       08:05:35.00 02/16/02*/
/*   INPUT    WSIM.IDCLOG                                            */
/*   OUTPUT   WSIM.STLFILE                                           */
/*   MSGTXT   STLDECK                                                */
/*   NODELAY                                                         */
/*   GENERATE ALL                                                    */
/*   LU       IDCSLU-1                                               */
/*   STL      TRACE=STLDECKT                                         */
/*   VERIFY   ROW=1,COLUMN=2,LENGTH=78                               */
/*------------------------------------------------------------------*/
panel_error = on                /* initialize panel error switch   */
do forever                      /* start do forever loop,          */
                                /* always leave after single pass  */

 onin0001: onin substr(ru,1,1) = 'F5'x,
               then found = on
 found = off
 initself('TSO01','NSX32702')
 do while found = off           /* wait for onin0001 data received */
  wait until onin
 end
 deact onin0001

 /* 07:58:29.36 ITP1507I SESSION STARTED WITH APPLICATION TSO01 */

 /* 07:58:31.83 ITP1508I SESSION ENDED WITH APPLICATION TSO01 */

 /* 07:58:31.99 ITP1507I SESSION STARTED WITH APPLICATION TSO0102 */

 screen_data = substr(screen,rowcol(1,2),78)
 expected_data = 'IKJ56700A ENTER USERID -'||repeat('00'x,54)
 verify screen_data ¬= expected_data for devid() msgtxtid() 'PNL00001',
                                        expected_data
 if     screen_data ¬= expected_data then leave
 cursor(1,27)
 ereof
 cursor(2,1)
 type 'user5'
 transmit using enter

 screen_data = substr(screen,rowcol(1,2),78)
 expected_data = '------------------------------- TSO/E  LOGON ---'||,
                 '------------------------------'
 verify screen_data ¬= expected_data for devid() msgtxtid() 'PNL00002',
                                         expected_data
 if     screen_data ¬= expected_data then leave
 transmit using enter

screen_data = substr(screen,rowcol(1,2),78)
 expected_data = 'IKJ56455I USER5 LOGON IN PROGRESS AT 07:58:43 ON'||,
                 ' FEBRUARY 16, 2002'||repeat('00'x,12)
 verify screen_data ¬= expected_data for devid() msgtxtid() 'PNL00003',
                                         expected_data
 if     screen_data ¬= expected_data then leave
 transmit using clear

screen_data = substr(screen,rowcol(1,2),78)
 expected_data = '-----------------------  ISPF/PDF PRIMARY OPTION'||,
                 ' MENU  -----------------------'
 verify screen_data ¬= expected_data for devid() msgtxtid() 'PNL00004',
                                         expected_data
 if     screen_data ¬= expected_data then leave
 cursor(2,15)
 ereof
 type '3.4'
 transmit using enter

screen_data = substr(screen,rowcol(1,2),78)
 expected_data = '--------------------------- DATA SET LIST UTILIT'||,
                 'Y -----------------------------'
 verify screen_data ¬= expected_data for devid() msgtxtid() 'PNL00005',
                                         expected_data
 if     screen_data ¬= expected_data then leave
 cursor(8,23)
 ereof
 type 'sys1'
 transmit using enter

 screen_data = substr(screen,rowcol(1,2),78)
 expected_data = 'DSLIST - DATA SETS BEGINNING WITH SYS1 ---------'||,
                 '----------------  ROW 1 OF 363'
 verify screen_data ¬= expected_data for devid() msgtxtid() 'PNL00006',
                                         expected_data
 if     screen_data ¬= expected_data then leave
 cursor(2,15)
 ereof
 type '=x'
 transmit using enter

 screen_data = substr(screen,rowcol(1,2),78)
 expected_data = 'READY '||repeat('00'x,72)
 verify screen_data ¬= expected_data for devid() msgtxtid() 'PNL00007',
                                         expected_data
 if     screen_data ¬= expected_data then leave
 cursor(1,9)
 ereof
 cursor(2,1)
 type 'logoff'
 transmit using enter
 /* 08:00:25.30 ITP1508I SESSION ENDED WITH APPLICATION TSO0102 */

 panel_error = off                /* no panel error, reset switch    */
 leave                            /* always leave do forever loop    */
end                               /* end do forever loop             */

verify panel_error = off for devid() msgtxtid() 'ALL PANELS VERIFIED'
if panel_error = on then
 do
  say devid() msgtxtid() 'PANEL VERIFICATION ERROR'
  suspend()                       /* panel error, suspend short time */
 end
screen_data = ''                  /* release storage for variables   */
expected_data = ''
endtxt
```

Below is an example of a message generation deck with panel verification.

```
WSIMDECK MSGTXT
*-----------------------------------------------------------------------
* ITPIDC: DISPLAY=WSIM420  APPLICATION=TSO01    07:58:23.89 02/16/02
*   ----------- DISPLAY CHARACTERISTICS AND FEATURES ------------
*   ALTCSET=APL               APLCSID=(963,310)
*   BASECSID=(697,37)         CCSIZE=(12,22)    COLOR=MULTI
*   DBCS=NO
*   DISPLAY=(24,80,24,80)     DLOGMOD=NSX32702  EXTFUN=YES
*   FLDOUTLN=NO               FLDVALID=NO       HIGHLITE=YES
*   MAXNOPTN=0                PS=NONE           UOM=INCH
*-----------------------------------------------------------------------
* ITPLSGEN: SCRIPT GENERATION PARAMETERS        08:05:35.00 02/16/02
*   INPUT    WSIM.IDCLOG
*   OUTPUT   WSIM.MSGFILE
*   MSGTXT   WSIMDECK
*   NODELAY
*   GENERATE ALL
*   LU       IDCSLU-1
*   WSim
*   VERIFY   ROW=1,COLUMN=2,LENGTH=78
*-----------------------------------------------------------------------
         CMND     COMMAND=INIT,RESOURCE=TSO01,MODE=NSX32702
0        IF       LOC=RU+0,TEXT=('F5'),
                  THEN=B-CONT0001
WAIT0001 WAIT
         BRANCH   LABEL=WAIT0001
CONT0001 DEACT    IFS=(0)

* 07:58:29.36 ITP1507I SESSION STARTED WITH APPLICATION TSO01

* 07:58:31.83 ITP1508I SESSION ENDED WITH APPLICATION TSO01

* 07:58:31.99 ITP1507I SESSION STARTED WITH APPLICATION TSO0102
         DATASAVE AREA=1,TEXT=($RECALL,(1,2),78$)
         DATASAVE AREA=2,TEXT=(IKJ56700A ENTER USERID -$DUP,00,54$)
         IF       LOC=1+0,LOCLENG=*,COND=NE,WHEN=IMMED,
                  TEXT=($RECALL,2$),
                  THEN=VERIFY-($DEVID$ $MSGTXTID$ PNL00001 $RECALL,2$),
                  ELSE=B-PNL00001
         BRANCH   LABEL=QUIT
PNL00001 LABEL
         CURSOR   ROW=1,COLUMN=27
         EREOF
         CURSOR   ROW=2,COLUMN=1
         TEXT     (user5)
         ENTER
         STOP

         DATASAVE AREA=1,TEXT=($RECALL,(1,2),78$)
         DATASAVE AREA=2,TEXT=(------------------------------- TSO/E),
                  (  LOGON ---------------------------------)
         IF       LOC=1+0,LOCLENG=*,COND=NE,WHEN=IMMED,
                  TEXT=($RECALL,2$),
                  THEN=VERIFY-($DEVID$ $MSGTXTID$ PNL00002 $RECALL,2$),
                  ELSE=B-PNL00002
         BRANCH   LABEL=QUIT
PNL00002 LABEL
         ENTER
         STOP

         DATASAVE AREA=1,TEXT=($RECALL,(1,2),78$)
         DATASAVE AREA=2,TEXT=(IKJ56455I USER5 LOGON IN PROGRESS AT ),
                  (07:58:43 ON FEBRUARY 16, 2002$DUP,00,12$)
         IF       LOC=1+0,LOCLENG=*,COND=NE,WHEN=IMMED,
                  TEXT=($RECALL,2$),
                  THEN=VERIFY-($DEVID$ $MSGTXTID$ PNL00003 $RECALL,2$),
                  ELSE=B-PNL00003
         BRANCH   LABEL=QUIT
PNL00003 LABEL
         CLEAR
         STOP

         DATASAVE AREA=1,TEXT=($RECALL,(1,2),78$)
         DATASAVE AREA=2,TEXT=(-----------------------  ISPF/PDF PRI),
                  (MARY OPTION MENU  -----------------------)
         IF       LOC=1+0,LOCLENG=*,COND=NE,WHEN=IMMED,
                  TEXT=($RECALL,2$),
                  THEN=VERIFY-($DEVID$ $MSGTXTID$ PNL00004 $RECALL,2$),
                  ELSE=B-PNL00004
         BRANCH   LABEL=QUIT
PNL00004 LABEL
         CURSOR   ROW=2,COLUMN=15
         EREOF
         TEXT     (3.4)
         ENTER
         STOP

         DATASAVE AREA=1,TEXT=($RECALL,(1,2),78$)
         DATASAVE AREA=2,TEXT=(---------------------------- DATA SET ),
                  (LIST UTILITY ----------------------------)
         IF       LOC=1+0,LOCLENG=*,COND=NE,WHEN=IMMED,
                  TEXT=($RECALL,2$),
                  THEN=VERIFY-($DEVID$ $MSGTXTID$ PNL00005 $RECALL,2$),
                  ELSE=B-PNL00005
         BRANCH   LABEL=QUIT
PNL00005 LABEL
         CURSOR   ROW=8,COLUMN=23
         EREOF
         TEXT     (sys1)
         ENTER
         STOP

         DATASAVE AREA=1,TEXT=($RECALL,(1,2),78$)
         DATASAVE AREA=2,TEXT=(DSLIST - DATA SETS BEGINNING WITH SYS),
                  (1 -------------------------  ROW 1 OF 363)
         IF       LOC=1+0,LOCLENG=*,COND=NE,WHEN=IMMED,
                  TEXT=($RECALL,2$),
                  THEN=VERIFY-($DEVID$ $MSGTXTID$ PNL00006 $RECALL,2$),
                  ELSE=B-PNL00006
         BRANCH   LABEL=QUIT
PNL00006 LABEL
         CURSOR   ROW=2,COLUMN=15
         EREOF
         TEXT     (=x)
         ENTER
         STOP
         DATASAVE AREA=1,TEXT=($RECALL,(1,2),78$)
         DATASAVE AREA=2,TEXT=(READY $DUP,00,72$)
         IF       LOC=1+0,LOCLENG=*,COND=NE,WHEN=IMMED,
                  TEXT=($RECALL,2$),
                  THEN=VERIFY-($DEVID$ $MSGTXTID$ PNL00007 $RECALL,2$),
                  ELSE=B-PNL00007
         BRANCH   LABEL=QUIT
PNL00007 LABEL
         CURSOR   ROW=1,COLUMN=9
         EREOF
         CURSOR   ROW=2,COLUMN=1
         TEXT     (logoff)
         ENTER
         STOP

* 08:00:25.30 ITP1508I SESSION ENDED WITH APPLICATION TSO0102

         IF       LOC=1+0,LOCLENG=1,TEXT=('00'),COND=GE,WHEN=IMMED,
                  THEN=VERIFY-($DEVID$ $MSGTXTID$ ALL PANELS VERIFIED)
         BRANCH   LABEL=END
QUIT     LABEL
         WTO      ($DEVID$ $MSGTXTID$ PANEL VERIFICATION ERROR)
         STOP
END      LABEL
         DATASAVE AREA=1,TEXT=()
         DATASAVE AREA=2,TEXT=()
         ENDTXT
```

Generating user delays

You may also create a script that reproduces the actual user delays that were recorded during data capture. For example, if you wait 10 seconds before entering data on a certain panel, when the script is executed, WSim waits 10 seconds before entering the data, too. If you do not request user delays, WSim enters the data when the default delay specified in the network definition expires.

You can select this option from either the IDC Generate STL Program panel (Figure 43 on page 191) or the IDC Generate Message Generation Deck panel (Figure 44 on page 195).

With this option, IDC generates a DELAY statement for each message sent in the script. The DELAY statement has a numeric value that tells WSim how long to wait before sending the next message to the host application.

How the value of the DELAY statement is computed depends on two things: the UTI (User Time Interval) and the think-time rule. You must specify both of these when you generate the script.

The UTI is expressed in hundredths of a second and for most scripts a value of 100 (1 second) is recommended. This value, multiplied by the value specified on the DELAY statement, gives the total amount of time WSim waits before sending the next message.

The think-time rule determines when WSim begins the countdown to send the next message. There are two choices: immediate and unlock. Immediate means that WSim begins counting the delay when it finishes building the previous message. Unlock means that WSim begins counting the delay when the 3270 keyboard is unlocked by the host application. For most situations, use unlock for the think time rule.

You must code the UTI and think-time rule in the network definition you eventually use when you run the script. Code the UTI value on the UTI operand of the NTWRK statement and code the think-time rule on the THKTIME operand of the DEV or LU statements. (Refer to the *WSim Script Guide and Reference* and *Creating WSim Scripts* for more information on coding these operands.) To help you remember what values you specified when you generate the script, IDC documents these values in the prologue it writes at the beginning of the STL program or message generation deck.

Here is an example of an STL program with delays.

```
@program=STLDECKT
STLDECK: msgtxt
/*-----------------------------------------------------------------*/
/* ITPIDC: DISPLAY=WSIM420  APPLICATION=TSO01    07:58:23.89 02/16/02*/
/*   ----------- DISPLAY CHARACTERISTICS AND FEATURES ------------   */
/*   ALTCSET=APL                APLCSID=(963,310)                    */
/*   BASECSID=(697,37)          CCSIZE=(12,22)    COLOR=MULTI        */
/*   DBCS=NO                                                         */
/*   DISPLAY=(24,80,24,80)      DLOGMOD=NSX32702  EXTFUN=YES         */
/*   FLDOUTLN=NO                FLDVALID=NO       HIGHLITE=YES       */
/*   MAXNOPTN=0                 PS=NONE           UOM=INCH           */
/*-----------------------------------------------------------------*/
/* ITPLSGEN: SCRIPT GENERATION PARAMETERS        08:05:35.00 02/16/02*/
/*   INPUT    WSIM.IDCLOG                                            */
/*   OUTPUT   WSIM.STLFILE                                           */
/*   MSGTXT   STLDECK                                                */
/*   DELAY    THKTIME=UNLOCK,UTI=100                                 */
/*   GENERATE ALL                                                    */
/*   LU       IDCSLU-1                                               */
/*   STL      TRACE=STLDECKT                                         */
/*   NOVERIFY                                                        */
/*-----------------------------------------------------------------*/

suspend()                         /* initial suspend to sync delays  */

onin0001: onin substr(ru,1,1) = 'F5'x,
               then found = on
found = off
delay(4)
initself('TSO01','NSX32702')
do while found = off             /* wait for onin0001 data received */
 wait until onin
end
deact onin0001


/* 07:58:29.36 ITP1507I SESSION STARTED WITH APPLICATION TSO01 */

/* 07:58:31.83 ITP1508I SESSION ENDED WITH APPLICATION TSO01 */

/* 07:58:31.99 ITP1507I SESSION STARTED WITH APPLICATION TSO0102 */


cursor(1,27)
ereof
cursor(2,1)
type 'user5'
delay(3)
transmit using enter


delay(3)
transmit using enter
```

```
delay(8)
transmit using clear


cursor(2,15)
ereof
type '3.4'
delay(6)
transmit using enter


cursor(8,23)
ereof
type 'sys1'
delay(6)
transmit using enter

cursor(2,15)
ereof
type '=x'
delay(4)
transmit using enter


cursor(1,9)
ereof
cursor(2,1)
type 'logoff'
transmit using enter

/* 08:00:25.30 ITP1508I SESSION ENDED WITH APPLICATION TSO0102 */

endtxt
```

Here is an example of a message generation deck with delays.

```
WSIMDECK MSGTXT
*----------------------------------------------------------------------
* ITPIDC: DISPLAY=WSIM420  APPLICATION=TSO01    07:58:23.89 02/16/02
*   ----------- DISPLAY CHARACTERISTICS AND FEATURES ------------
*   ALTCSET=APL                APLCSID=(963,310)
*   BASECSID=(697,37)          CCSIZE=(12,22)    COLOR=MULTI
*   DBCS=NO
*   DISPLAY=(24,80,24,80)      DLOGMOD=NSX32702  EXTFUN=YES
*   FLDOUTLN=NO                FLDVALID=NO       HIGHLITE=YES
*   MAXNOPTN=0                 PS=NONE           UOM=INCH
*----------------------------------------------------------------------
* ITPLSGEN: SCRIPT GENERATION PARAMETERS        08:05:35.00 02/16/02
*   INPUT    WSIM.IDCLOG
*   OUTPUT   WSIM.MSGFILE
*   MSGTXT   WSIMDECK
*   DELAY    THKTIME=UNLOCK,UTI=100
*   GENERATE ALL
*   LU       IDCSLU-1
*   WSIM
*   NOVERIFY
*----------------------------------------------------------------------


         CMND     COMMAND=INIT,RESOURCE=TSO01,MODE=NSX32702
         DELAY    TIME=4
0        IF       LOC=RU+0,TEXT=('F5'),
                  THEN=B-CONT0001
WAIT0001 WAIT
         BRANCH   LABEL=WAIT0001
CONT0001 DEACT    IFS=(0)
```

```
* 07:58:29.36 ITP1507I SESSION STARTED WITH APPLICATION TS001

* 07:58:31.83 ITP1508I SESSION ENDED WITH APPLICATION TS001

* 07:58:31.99 ITP1507I SESSION STARTED WITH APPLICATION TS00102


         CURSOR   ROW=1,COLUMN=27
         EREOF
         CURSOR   ROW=2,COLUMN=1
         TEXT     (user5)
         ENTER
         DELAY    TIME=3


         ENTER
         DELAY    TIME=3


         CLEAR
         DELAY    TIME=8


         CURSOR   ROW=2,COLUMN=15
         EREOF
         TEXT     (3.4)
         ENTER
         DELAY    TIME=6


         CURSOR   ROW=8,COLUMN=23
         EREOF
         TEXT     (sys1)
         ENTER
         DELAY    TIME=6


         CURSOR   ROW=2,COLUMN=15
         EREOF
         TEXT     (=x)
         ENTER
         DELAY    TIME=4


         CURSOR   ROW=1,COLUMN=9
         EREOF
         CURSOR   ROW=2,COLUMN=1
         TEXT     (logoff)
         ENTER

* 08:00:25.30 ITP1508I SESSION ENDED WITH APPLICATION TS00102

         ENDTXT
```

Modifying IDC-generated scripts

Once you create a script, you are free to modify it. You may want to do this for several reasons. Usually, you do this to allow the script to be used by multiple devices in the simulated network at the same time.

For example, suppose you want several simulated devices to all log on to TSO simultaneously. Each device needs to use a different TSO user ID. You could create a different script for each device, with each script referencing a different user ID. A better solution is to place all of the user IDs in a WSim user table and replace the actual user ID in the script with a reference to an entry in the table. Each time you execute the script, it selects a different entry from the table.

Other reasons to modify a script might be to provide error recovery actions, coordinate actions between simulated devices with signals and events, or add iterative or conditional logic to control the execution of the script.

Using help

You can get help on all IDC panels, fields and selection options. The type of help you get depends on the panel you are on and where the cursor is when you press **F1**.

1. If you are currently on an IDC panel and the cursor is on a data entry field or selection option, specific help for that field or option is displayed.
2. If you are currently on an IDC panel and the cursor is not on a data entry field or selection option, general help for the panel is displayed.
3. If you are on a field or selection option help panel, general help for the panel is displayed.

Debugging problems

If you have a problem with the capture process or the generated script, there are two sources of information that may help: the IDC log and the IDC trace.

If your script didn't work the way you expected, the IDC log can help you understand what was sent and received by the display during data capture. The IDC log shows all of the data flows from the 3270 display's point of view. Comparing the IDC log to the WSim log will show any differences between the data capture session and the WSim simulation session.

If you have a problem during data capture, the IDC trace can help. The IDC trace shows all of the data flows from IDC's point of view. This includes data from both IDC sessions: the session with the 3270 display and the session with the host application.

Analyzing the IDC log

The records IDC writes in its log data set are in the same format that WSim uses for its log data set. (The WSim log data set provides a record of the simulation run.) Therefore, the Loglist Utility, which is used to analyze the WSim log, can also be used to analyze the IDC log. Consult Part 1, “General utilities,” on page 1. for instructions on how to use the Loglist Utility.

IDC writes receive (RECV), transmit (XMIT), and 3270 log display (DSPY) records to the IDC log while it is capturing data. IDC writes these records using “ITPIDC” as the network and message generation deck names, “IDCVA” as the VTAMAPPL name, “IDCSLU-1” as the LU name, and “E2” (LU type 2) as the terminal type.

Receive (RECV) and Transmit (XMIT) Records
IDC creates receive and transmit records directly from the SNA data traffic flows between itself and the host application program while it's capturing data. These SNA data traffic flows represent the actual flows a 3270 display exchanges with the host application program.

3270 Log Display (DSPY) Records
IDC creates 3270 log display records to reflect the screen images being

displayed on the 3270 display being used. IDC creates “BEGINNING OF MSG GEN” and “END OF MSG GEN” log display records when it receives data from the 3270 display.

Running IDC with the DEBUG execution parameter (see “Specifying execution parameters” on page 179) provides a way to correlate generated statements in the script to captured data in the IDC log.

Analyzing the IDC trace

Errors can occur between the host application and the display. A severe error causes IDC to terminate its session with the host or the display. To analyze a failure from IDC's point of view, you must start IDC with the “TRACE” execution parameter (see “Specifying execution parameters” on page 179) and code a DD or FILEDEF statement for the IDC trace data set. Next, you need to recreate the problem. The IDC trace contains records for the session between the 3270 display and IDC and the session between IDC and the host application. The records in the IDC trace are in the same format as the IDC log and the WSim log, so you can use the Loglist Utility to analyze the IDC trace. See Part 1, “General utilities,” on page 1. for instructions on how to use the Loglist Utility.

The IDC trace contains the same record types as the IDC log. IDC writes these records using “ITPIDC” as the network and message generation deck names, “IDCVA” as the VTAMAPPL name, “IDCPLU-1” as the IDC PLU name, “IDCSLU-1” as the IDC SLU name, and “E2” (LU type 2) as the terminal type.

Understanding IDC restrictions

Keep the following items in mind when you run IDC.

- Single 3270 display usage

  IDC only supports a single 3270 display session. It rejects the logon request if a 3270 display session already exists.
- Only DASD IDC log data sets are supported
- Altering the query response

  Host application programs normally query the 3270 display to determine the extended functions supported. IDC alters the query response from the 3270 display to only indicate support of the 3270 functions that can be simulated by WSim and passes the altered query response to the host application program.
- Unsupported 3270 functions

  IDC only supports 3270 display functions that can be simulated by WSim. IDC notifies you whenever a 3270 data stream using unsupported 3270 display functions is received and sends a negative response to the sender.

  Specifically, WSim and IDC support the following 3270 extended functions:
  - Color
  - Highlighting
  - Character sets for base, APL, and up to six Programmed Symbols (PS)
  - Double-byte character set (DBCS)
  - Field validation
  - Alphanumeric partitions
  - Field outlining.

  Among the 3270 extended functions not supported by WSim and IDC are:
  - File transfer

  - All points addressable (APA) graphics
  - Auxiliary devices.
- Non-3270 display usage

  IDC only supports 3270 displays. IDC rejects the logon request if the BIND image in the CINIT request is not a valid 3270 LU type 2 or LU type 0 BIND image. Also, the type of BIND image (LU type 2 or LU type 0) must be the same for the 3270 display and application program sessions with IDC.
- Script generating assumptions

  IDC-generated scripts assume all data can be entered by the simulated operator during WSim message generation. Data in application-initialized fields may cause WSim to log informational messages during message generation. You may need to modify your script if this happens.
- Generating Network Services and Data Flow Control statements

  IDC generates statements to simulate INIT-SELF, TERM-SELF, LUSTAT, and SIGNAL flows only.
- Panel verification using multiple partitions

  When IDC captures data from an application program that defines multiple alphanumeric partitions and panel verification logic is generated into the script, only the panel data in one of the defined partitions can be verified.

  The panel verification logic row and column values are used to locate the partition owning the viewport at that location on the 3270 screen. The logic test is made against the partition presentation space data starting at that location for the specified length.

  Because of this, you should specify a location that contains predictable information, such as a panel identifier.
- Window location using scrollable partitions

  When IDC captures data from an application program that defines scrollable alphanumeric partitions, IDC cannot maintain the exact partition window location within the partition presentation space, because this information is not available to the program. IDC always moves the window to include the cursor, which may or may not display the same window data as seen on the 3270 display. Because of this, the log display records created by IDC may not reflect the actual displayed data. However, the generated script does accurately reflect the data entered when used for the WSim simulation run.
- Limited field validation trigger support

  You must modify the scripts generated by IDC for data entry into field validation trigger fields to move the cursor out of a primed trigger field. You can normally do this with the tab or new line key; however, this information is not available to IDC. You should insert the cursor movement statement following the “Trigger Field Data Generated” comments in the generated script.
- No RESET key support

  The ITPIDC or ITPLSGEN generated script assumes the simulated 3270 display keyboard is unlocked and the SNA state allows the next message to be generated. If your application requires the RESET key and optionally the ATTN key to be pressed before sending the next message, you must insert logic into the generated script to perform a RESET key operation to unlock the keyboard and optionally generate an SNA SIGNAL request to simulate the ATTN key operation.
- TSO console support

  When you run IDC with the TSOCON execution parameter, the following restrictions apply:

- If you run IDC in a local LU type 0 environment, be aware of the following:
  - The Read Modified All command is not supported. If your application issues this command, you see the following message: “IKT00405I Screen erasure caused by error recovery procedure”.
  - Your display screen may flash when non-display commands are issued to your terminal. The amount of flashing depends on the type of display. Gas panel displays tend to flash the most.
- Pressing the attention key (ATTN or PA1) does not send an SNA SIGNAL command to your application. If your application requires a key to generate this command, run IDC without specifying TSOCON.
- If you run IDC in an LU Type 2 environment, IDC may unlock your keyboard earlier than expected. Some applications handle type-ahead input better than others. When you are in session with an application that is sensitive to type-ahead input, use caution to avoid entering data before the application expects it.
- Terminal communications from other users or the TSO operator destroys the currently displayed panel. When you finish reading these messages, you must press **Clear** twice to redisplay the panel. If your application uses a specific reshow key, use it instead of **Clear**.

  To minimize this, set your TSO profile to NOINTERCOM before starting IDC with the TSOCON execution parameter. This option prevents your terminal from receiving other terminal users' messages. However, you still receive high priority messages from the TSO operator. The IDC TSO CLIST for running with the TSOCON execution parameter sets the profile to NOINTERCOM and resets the original environment when IDC completes. (See “Running IDC from a CLIST” on page 179 for an example CLIST.) If you want to receive messages from other users while running IDC, modify this CLIST accordingly. If you run IDC from the WSim/ISPF Interface, you can specify Y or N in the Suppress user messages field of the Interactive Capture and Build Message Decks and STL Programs panel. All message activity is reflected in the generated scripts.
- IDC SYSPRINT output should not be routed to the TSO console. Doing so causes undesired effects in the tracing activity and resulting scripts. The IDC TSO CLIST for running with the TSOCON execution parameter allocates SYSPRINT to the print queue. If this command is removed from the CLIST, SYSPRINT may be routed to the TSO console, depending on your TSO system definition.

Creating network definitions

In addition to generating an STL program or message generation deck, you also need to create a WSim network definition to run your script. For information on this task, refer to *Creating WSim Scripts* and *WSim Script Guide and Reference*.

Here is a example of a VTAMAPPL network definition for a single 3270 display.

```
SAMPNET   NTWRK     HEAD='VTAMAPPL 3270 NETWORK',
                    BUFSIZE=2048,
                    UTI=100,
                    THKTIME=UNLOCK
PATH1     PATH      STLDECK
WSIMAPPL  VTAMAPPL
T3270A    LU        INIT=SEC,
                    LUTYPE=LU2,
                    ALTCSET=NONE,
                    CCSIZE=(9,16),
                    COLOR=GREEN,
                    DISPLAY=(24,80,24,80),
                    DLOGMOD=NSX32702,
                    EXTFUN=YES,
                    FLDVALID=NO,
                    HIGHLITE=NO,
                    PS=NONE,
                    UOM=INCH
```

The network you define must match the script you generated in the following ways:

1. The PATH statement must reference the STL program or message generation deck name you specified when you generated the script.
2. INIT=SEC must be coded to indicate that the 3270 display (the SLU) initiates the session. If the script does not include a generated logon sequence (INITSELF or CMND COMMAND=INIT), you can code the RESOURCE operand and let WSim establish the session before executing the script.
3. If you requested user delays when you generated the script, you should code the same UTI and THKTIME values in the network definition.
4. The display characteristics and features of the simulated 3270 should be compatible with the actual 3270 display you used when you captured the data, such as screen size and color support. The prologue in the generated script shows the operands describing the actual 3270 display.

Controlling the flow of script execution

The WSim message decks or STL programs you specify on the PATH statement are executed repeatedly until you cancel the network or stop WSim. If you want the decks or programs to be executed only once, you can add a message deck to the end of PATH statement that executes a QUIESCE statement.

In STL, the deck would look like this:

```
qdeck: msgtxt
       say 'Device' luid() 'now quiesced.'
       quiesce
       endtxt
```

In the WSim scripting language, the deck would look like this:

```
QDECK   MSGTXT
        WTO (Device  $LUID$ now quiesced.)
        QUIESCE
        ENDTXT
```

There other ways to control flow of execution of decks and programs on the PATH statement (random selection and selection according to a percentage distribution) that require special network definition statements. For more information on these options, refer to *Creating WSim Scripts*.

Synchronizing multiple scripts

If you plan to have a single simulated 3270 device execute multiple scripts in sequence, you need to plan the creation of the scripts carefully. Specifically, if script A is to follow script B, then B must begin at the panel where A ends. One way to ensure this is to have all your scripts begin and end at the same host application panel. Another way is to have each script logon and logoff the application separately.

You also need to add a delay between each script on the PATH statement. This is because the next deck or program on the PATH statement is executed immediately after the last. There are several ways to do this.

1. Add a delay statement to the end of each generated script.
2. Create a message deck or STL program similar to the examples in “Controlling the flow of script execution” on page 210 (but with a DELAY statement instead of a QUIESCE) and specify it between each deck or program on the PATH statement.
3. Write a “main” deck or program that calls the other decks or programs. This “main routine” controls the sequence and timing of the execution of the scripts.

Understanding IDC return codes

At the end of execution, IDC sets a return code to indicate the status of the execution. IDC execution ends prematurely for all return codes except 0. IDC can return the following codes:

Code	Meaning
0	Execution ended successfully.
4	Execution was attempted under something besides MVS or TSO.
8	A parameter error was detected.
12	An error occurred OPENing the ACB.
16	An error occurred issuing the VTAM SETLOGON request.
20	A GETMAIN storage request failed.
24	VTAM scheduled the TPEND exit routine.

Running WSim

When the script and network definition are complete, you are ready to run WSim. For information on running WSim, see *WSim User's Guide*.

Generating Telnet 3270 scripts

The generated VTAMAPPL 3270 scripts can be converted into Telnet 3270 scripts.

The major difference in using the TCP/IP support versus the SNA support is in establishing the session. Instead of using the CMND= or RESOURCE= operands, you will need to code a FRSTTXT= message generation deck to establish the session with the application logon screen. After the session is established, the scripts should be the same.

You will need the decide if you want to use the WSim Telnet 3270 or 3270E support. This will depend on what type of server you want to test. If you are not sure, you should use the 3270E.

Listed below are the steps that may or may not need to be executed.

These are the main VTAMAPPL operands that do not have meaning for Telnet 3270 simulation and should be removed:

- INIT=
- LUTYPE=
- DLOGMOD=

- MAXSESS=
- CMND=

Instead of VTAMAPPL and LU, you will need to code something like the following:

```
WSIMAPPL  TCPIP   TNPORT=23                       Telnet 3270 default port

TERM01            DEV DISPLAY=(24,80,32,80),      Primary and alternate display size
                      EXTFUN=YES,
                      LOGDSPLY=BOTH,
                      SERVADDR=xxx.xxx.xxx.xxx,   IP Address
                      TYPE=TN3270E                Telnet 3270E

TERM02            DEV ...
```

See the *WSim User's Guide* for more information on the operands.

Chapter 16. Generating scripts from IDC or WSim log data sets

You can also generate STL programs or WSim message generation decks from an IDC or WSim log data set with the Log Script Generator Utility (ITPLSGEN). ITPLSGEN generates a script in the same format as the Interactive Data Capture (IDC) utility ITPIDC. ITPLSGEN lets you generate or re-generate, using different script generation parameters, an STL program or WSim message generation deck outside of the IDC environment. You can run ITPLSGEN under MVS and control it by using an input file or operator entered commands.

ITPLSGEN also lets you generate an STL program or WSim message generation deck from a WSim log data set created during a WSim simulation run. ITPLSGEN generates the script from pairs of the simulated 3270 “BEGINNING OF MSG GEN” and “END OF MSG GEN” log display records along with the transmit data records on the WSim log data set. If the WSim log data set contains all of these records, you can use ITPLSGEN to create a new script, which may be useful in some environments. For example, you can generate an STL program with a WSim log data set containing records created from the execution of an old WSim message deck. You can then use this program as the base for a new STL program to perform a similar function.

In order to successfully generate a script from a WSim log data set, you must meet the following conditions when you create the WSim log. These requirements apply only to the simulated 3270 device from which the script is generated. They do not apply to any other resources in the network.

- The LU name must be unique within the VTAMAPPL.
- LOGDSPLY=BOTH must be specified.
- MLOG=YES must be specified if you want to generate user delays.

Setting up ITPLSGEN

To run ITPLSGEN under MVS, you need to allocate or use an existing allocation for the following data sets:

STL programs
This data set contains the generated STL programs. If you plan to generate STL programs, allocate this as a fixed block, variable block, or variable data set with a record length of at least 71 bytes and a block size compatible with the record length. You may allocate this as either a partitioned or sequential data set. The space needed for this data set depends on the number of user actions and amount of data generated in the program. Initially, allocate at least 5 cylinders of 3390 DASD or equivalent space for this data set. This data set must be cataloged.

WSim message generation decks
This data set contains the generated WSim Scripting Language message generation decks. If you plan to generate message generation decks, allocate this as a fixed block data set with a record length of 80 bytes and a block size compatible with the record length. You may allocate this as either a partitioned or sequential data set. The space needed for this data set depends on the number of user actions and amount of data generated in the message

generation decks. Initially, allocate at least 5 cylinders of 3390 DASD or equivalent for this data set. This data set must be cataloged.

`ITPLSGEN commands`
This data set contains the ITPLSGEN commands used to control the script generation process. Allocate this as a fixed block data set with a record length of 80 bytes and a block size compatible with the record length. The space needed for this data set depends on the number of ITPLSGEN commands entered.

Remember, the sizes given above are just estimates to get you safely started. You should base any long range predictions for space requirements on expected use and actual experience with ITPLSGEN.

Running ITPLSGEN

Once you set up the data sets, you can then start the ITPLSGEN utility. To do this, you need to define the ITPLSGEN job stream and specify the ITPLSGEN execution parameters. You can also run ITPLSGEN by way of the WSim/ISPF Interface. The following topics discuss how to run ITPLSGEN.

Using JCL to run ITPLSGEN

Below shows the JCL to run ITPLSGEN as an MVS batch job.

```
//ITPLSGEN JOB
//ITPLSGEN EXEC     PGM=ITPLSGEN
//STEPLIB  DD       DSN=WSIM.SITPLOAD,DISP=SHR
//SYSPRINT DD       SYSOUT=A
//SYSUT1   DD       UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSIN    DD       DSN=WSIM.ITPLSGEN.COMMANDS,DISP=SHR
```

Using a CLIST to run ITPLSGEN

The example below shows the CLIST commands to run ITPLSGEN under TSO.

```
ALLOC    DDNAME(SYSPRINT) SYSOUT(A)
ALLOC    DDNAME(SYSUT1)   UNIT(SYSALLDA) SPACE(1,1) CYL
ALLOC    DDNAME(SYSIN)    DATASET('WSIM.ITPLSGEN.COMMANDS') SHR
CALL     'WSIM.SITPLOAD(ITPLSGEN)'
FREE     DDNAME(SYSPRINT SYSUT1 SYSIN)
```

Running ITPLSGEN from the WSim/ISPF Interface

To invoke ITPLSGEN from the WSim/ISPF Interface, follow these steps:

1. Invoke the WSim/ISPF Interface main panel from ISPF. The method you use to do this depends on how the WSim/ISPF Interface application is installed at your site. If you are not sure how to do this, see your system programmer for assistance.
2. Select option 5 from the WSim/ISPF Interface main panel and press **Enter**. The Generate Message Decks and STL Programs panel is displayed.
3. Select option 1 from this panel and press **Enter**. The Generate Message Decks and STL Programs from Log Data Set panel is displayed.

   Note: You can also type "LSGEN" on the WSim/ISPF Interface main panel command line and press **Enter** to display this panel.
4. Fill in the appropriate fields on this panel and press **Enter** to run ITPLSGEN.

For more information about the WSim/ISPF Interface, refer to Part 1, "General utilities," on page 1.

Specifying execution parameters

You can specify the following execution parameters for ITPLSGEN:

`CONSOLE`
: Tells ITPLSGEN to issue write-to-operator-with-reply (WTOR) messages to the operator console for the input control commands. If you do not specify this parameter, ITPLSGEN reads control commands from the SYSIN data set.

`PRTLNCNT=`*`nnn`*
: Specifies the maximum number of lines that can be printed on a page of SYSPRINT output before ejecting to a new page. *nnn* can be an integer from 35 to 255. The default value for *nnn* is 60.

`ROUTCDE=(`*`n`*`,`*`n`*`,...)`
: Specifies the system message routing codes to be used when IDC writes messages to the operator. Each *n* is a system routing code that defines a console destination for every WTO and WTOR message ITPLSGEN writes. *n* can be an integer from 1 to 16. The default value for the ROUTCDE parameter is 8.

Using control commands

You can enter control commands to complete the following tasks:

- Define the program name or message generation deck name
- Specify the input and output data set names
- Generate an STL program or a WSim message generation deck
- Define user delays
- Generate changed data fields
- Define panel verification logic parameters
- Enter commands from the input data stream
- Generate debugging comments
- Start the utility
- End the utility

Note: You must enter the following input commands:

- MSGTXT
- INPUT
- OUTPUT
- RUN
- END.

All other commands are optional.

Entering control commands

You can enter a control command in any position in the input record of the SYSIN DD data set. These commands cannot extend past column 71 and cannot be continued. Operands can only be separated with commas. Although you can separate a command and its operand by more than one blank space, you must enter at least one blank space between the command and the operand. You cannot enter blank spaces between operands; operands that follow the space are interpreted as comments. To enter a comment on a line with operands, insert at least one space in between the operands and the comment.

You might also have comment lines. You can do this by entering an asterisk (*) in the first column.

If you enter a command more than once within the same run, ITPLSGEN uses the last valid one entered. You can enter multiple RUN commands in the SYSIN DD data set or from the WSim operator console. Successful generation of a script following a RUN command clears all the commands entered; a RUN command that results in an error does not clear the commands.

All parameters, commands, operands, and some operand values [2] can be abbreviated to any shorter length. All of these, except for NOVERIFY, can be shortened to one letter. The abbreviated command for NOVERIFY is NOV. The following examples are identical.

```
VERIFY ROW=3,COLUMN=5,LENGTH=7
V R=3,C=5,L=7
VER R=3,COLUMN=5,LENG=7
```

Defining the STL program or message generation deck name

You define the name of the generated STL program or message generation deck with the MSGTXT command. The syntax of this command is shown below.

MSGTXT *msgtxtid*

msgtxtid must be 1 to 8 alphanumeric or special ($,@,_,?,#) characters, where the first character is non-numeric.

For an STL program, the name cannot be an STL reserved word or begin with $INC, $LA, or $SET (reserved labels in STL). Also, you cannot use the same name as the STL program trace name.

Specifying input and output

You can specify the input and output data sets by using the INPUT and OUTPUT commands.

To specify the input data set name, code the INPUT command. This command is shown below.

INPUT *dsname*

dsname is the data set name of the IDC or WSim log data set. This name can be from 1 to 36 characters long and can be enclosed in single or double quotes. If the data set name you specify has imbedded spaces in it, you must enclose the name in quotes. When you are using partitioned data sets under MVS, specify the member name within parentheses after *dsname*.

To specify the output data set name, code the OUTPUT command. This command is shown below.

OUTPUT *dsname*

dsname is the data set name containing the generated script. This name can be from 1 to 36 characters long and can be enclosed in single or double quotes. If the data set name you specify has imbedded spaces in it, you must enclose the name in

2. IMMED and UNLOCK only.

quotes. When you are using partitioned data sets under MVS, the member name may optionally be specified within parentheses after *dsname*. The default member name for output partitioned data set is the *msgtxtid* specified on the MSGTXT command.

Choosing the type of script to generate

You may generate either an STL program or a WSim message generation deck from the IDC or WSim log data set. You can do this using the STL and WSim control commands.

To generate an STL program from the log data set, use the STL control command. This command is shown below.

STL [TRACE=*progname***]**

The optional TRACE operand lets you specify the name you want to appear on the STL @PROGRAM statement, which is used to define a symbolic name for labeling STL trace records. *progname* must be 1 to 8 alphanumeric or special ($,@,_,?,#) characters, where the first character is non-numeric. The name cannot be an STL reserved word or begin with $INC, $LA, or $SET (reserved labels in STL). Also, you cannot use the same name as the MSGTXT.

You only need to use the TRACE operand if you want to trace the execution of your STL program at run time.

To generate a WSim message generation deck, use the WSim control command. This command is shown below.

WSIM

If you do not specify either the STL or WSim control commands, ITPLSGEN generates an STL program.

Specifying devices

You can specify the device in the WSim log data set from which to generate the script by using the LU command. This command is shown below.

LU *name***[-***nnnnn***]**

The LU command specifies which LU in the WSim log data set the utility uses to generate the script. *name* is a 1 to 8 character name and must be an actual LU or DEV statement from the network that generated the WSim log data set. *nnnnn* is a 1 to 5 digit number and must be an actual LU session number from the network that generated the WSim log data set.

If you do not use this command, ITPLSGEN uses the default name IDCSLU-1, which is the LU name and session number for an IDC generated log.

Note: The LU name, in combination with the network name and the VTAMAPPL name, must be unique in records from the WSim log data set. If your WSim log has a network name of ITPLU2RF, indicating it was produced by the SNA 3270 Reformatter Utility, the LU name must be unique in combination with the network name.

Defining user delays

You can reproduce actual delay times recorded during the IDC capture or WSim run, representing user think time delays, with the DELAY control command. You can also disable delays with the NODELAY command.

To use actual user delays, enter the DELAY command. This command generates DELAY statements in the generated script that reproduce actual user delays. (Refer to *Creating WSim Scripts* for a discussion of intermessage delays.) This command is shown below.

DELAY [THKTIME={IMMED|UNLOCK}] [,UTI=*nnnn*]

The optional THKTIME operand lets you specify what think time rules to use in calculating the delays. You can enter either IMMED to use immediate think time rules or UNLOCK to use unlock think time rules. If you do not specify the THKTIME operand, the generated script calculates user delays using the unlock think time rules.

The optional UTI operand lets you specify the user time interval value (in hundredths of a second) that the generated script uses to calculate delays. This must be a number between 1 (representing .01 seconds) and 6000 (representing 60 seconds). If you do not specify the UTI operand, the generated script uses a user time interval of 100 (one second). (Refer to *WSim Script Guide and Reference* for more information on coding the THKTIME and UTI operands.)

Note: If you want to simulate LU type 2 devices, use THKTIME=UNLOCK here and in your network definition.

If you do not want to generate user delays, omit the DELAY command or enter the NODELAY command.

NODELAY

If you do not specify either the DELAY or NODELAY command, ITPLSGEN generates a script without user delays.

Generating changed data fields

You can select whether to generate scripts with all data fields on your application's panels or only those that you actually changed. To do this, use the GENERATE command, shown below.

GENERATE ALL|CHANGED

Use GENERATE ALL if you want IDC to generate scripting statements for all the unprotected fields on the screen containing data sent to the application program. This includes unprotected fields set by the application with the MDT bit on that you did not change during the data capture session.

Use GENERATE CHANGED if you want IDC to generate scripting statements for only the unprotected fields that you actually changed during the data capture session. Unprotected fields set by the application with the MDT bit on are ignored for script generation.

Note: Unprotected fields with the 3270 Modified Data Tag (MDT) bit set on are sent to the application program when you press Enter or a function key.

If you do not specify the GENERATE command, ITPLSGEN generates statements for all data fields on the application's screens.

Verifying panels

You can generate scripts with logic to check each panel in the IDC or WSim log data set for particular data at a given location, such as a panel identifier. To do this, use the VERIFY command, shown below.

VERIFY ROW=*nnn***,COLUMN=***nnn***,LENGTH=***nnn*

When you enter the VERIFY command, ITPLSGEN generates a script that checks each panel at the given row and column for the given length against the panels stored in the log data set at the same row and column for the same length.

The ROW and COLUMN operands specify the starting location of the string to check on each panel. *nnn* can be a number from 1 to 255. The LENGTH operand specifies the length of the data to be verified on the panel at the given location. *nnnnn* can be a number from 1 to 32000.

The amount of data verified can cross rows. For example, to verify the second and third rows on an 80-column screen, specify a starting location of row 2, column 1, with a length of 160. If you specify a length that exceeds the screen size, the length value is reduced to match the size.

Note: You must code all three operands on the VERIFY command.

To generate scripts without panel verification, omit the VERIFY command or use the NOVERIFY command, shown below.

NOVERIFY

If you do not specify either the VERIFY or NOVERIFY command, ITPLSGEN generates a script without panel verification logic.

Entering commands from the input data stream

You can use the P control command to terminate input from the WSim operator console and begin reading commands from the SYSIN data set, as follows:

P

If you enter the P command and the SYSIN data set is not open, the utility continues requesting input from the console. This command is ignored if it is encountered in the SYSIN data stream.

Generating debugging comments

You can use the DEBUG command to generate debugging comments as follows:

DEBUG

When you enter the DEBUG command, ITPLSGEN inserts comments into the generated script before each group of generated statements. Each comment indicates the time of day and sequence number of the record in the IDC or WSim log data set used to generate the statements following the comment.

A sample comment for STL programs is shown below.

```
/*--------------------------------------------------- 07582389 00001 */
```

A sample comment for WSim message generation decks is shown below.

```
*------------------------------------------------------ 07582389 00001
```

Use the DEBUG command as a debugging aid for script generation. If you do not specify the DEBUG command, ITPLSGEN does not generate debugging comments.

Starting ITPLSGEN

When you finish entering your control commands, you then start the ITPLSGEN utility with the RUN command. This command is shown below.

RUN

The RUN command specifies that all commands are entered and that script generation is to begin. After script generation, ITPLSGEN resets everything to their default values.

You can only enter RUN after you specify MSGTXT, INPUT, and OUTPUT for that run. If you enter RUN before specifying the above commands, ITPLSGEN does not process any of your commands. This lets you enter the required commands without losing what you previously entered.

Ending ITPLSGEN

The last control command you enter is the END command. This command is shown below.

END

The END command tells ITPLSGEN to come to a normal completion. No further processing occurs. Any commands entered since the last RUN command are ignored.

Generating the output

ITPLSGEN generates either an STL program or a WSim Scripting Language message generation deck, depending on whether you entered the STL or WSim control command. Examples of generated scripts are shown in “Generating scripts” on page 191.

Chapter 17. Generating 3270 scripts from captured traces

The SNA 3270 Reformatter Utility (ITPLU2RF) is a WSim utility that reformats session collection data logged by NPM or VTAM. This session collection data is known to NPM as the VTAM PIU log (FNMVLOG). The SNA 3270 Reformatter Utility processes this NPM log to create an equivalent WSim log data set for LU type 2 sessions or LU type 0 sessions using a 3270 data stream (that is, sessions between applications and 3270 terminals). It also produces reports showing statistics on record counts, data lengths, and time stamps.

Note: You must use NPM Version 1 Release 4 or later, or VTAM Version 4 Release 1 or later, Full Buffer Trace, to capture logs to be reformatted by the SNA 3270 Reformatter Utility.

You can use NPM to trace one or more users, or even an entire application, without any of the users knowing they are being traced. This is similar to IDC, where a single user can be traced to create an IDC log. However, this user must first log on to IDC and so cannot be unknowingly traced.

You can use the SNA 3270 Reformatter Utility along with the WSIM Log Script Generator Utility (ITPLSGEN) to create STL programs and WSim message decks. This contrasts with the Script Generator Utility (ITPSGEN), which only creates WSim message decks without the additional 3270 support unique to ITPLSGEN. You still may prefer to use ITPSGEN, as it can process other LU types as well as LU type 2.

The WSim log data set created by the SNA 3270 Reformatter Utility can be used by other WSim utilities such as Loglist, Response Time, and Log Compare. For example, you could process a reformatted log through the Log Script Generator to create a WSim script. This WSim script could then be used when you run your simulation to create a WSim log data set. You use the Log Compare Utility to compare the WSim log with the actual production run logged by NPM or VTAM and reformatted by the SNA 3270 Reformatter Utility. Your comparison report compares the actual production run with the first test run of the WSim script. You can document the results of your script execution by running the Loglist Utility. This prints the screen images from the WSim log data set created by the SNA 3270 Reformatter Utility. Similarly, you can use the Response Time Utility to measure the actual production response times by processing the reformatted NPM log.

Planning to use the SNA 3270 Reformatter Utility

Before collecting data, you must plan what you want to trace. In particular, you need to determine what terminals and applications you need for reformatting and whether you need to record the logmode for each session. The logmode is included in STL programs or WSim message decks when a WSim log data set is processed by the Log Script Generator (ITPLSGEN). This lets you specify the same logmode as traced by for this session when you actually run your simulation. If this logmode is important to you, you must trace the application, otherwise you may trace specific terminals. Tracing specific terminals ensures you only collect data for sessions you want to reformat.

For the SNA 3270 Reformatter Utility to reformat sessions properly, you need a BIND record in the log data set for each session to be reformatted. Any sessions that exist when you start the trace will not have BIND records in the log data set and are not reformatted. To make sure that BIND records are present for the desired sessions, start the trace before you establish any sessions.

NPM Version 1 Release 4 or later must be properly installed on your system before you can trace any sessions to be reformatted by the SNA 3270 Reformatter Utility if you use NPM traces as input. Refer to *NPM Installation and Customization* for more information on installing NPM.

Refer to *NPM Operation* for instructions on capturing traces using NPM.

Allocating SNA 3270 Reformatter Utility data sets

Before you run the SNA 3270 Reformatter Utility, allocate three data sets: the log for input (as described above), the WSim log data set, and SYSPRINT for output.

The space needed for the WSim log data set depends on the size of the input log. Initially, allocate at least 150 blocks (5 cylinders of 3380 DASD or equivalent) for this data set. For 3380 DASD, allocate this data set as a variable block, sequential or partitioned data set with a record length of 23472 bytes and a block size of 23476 bytes. For 3390 DASD, allocate this data set as a variable block, sequential or partitioned data set with a record length of 27994 bytes and a block size of 27998 bytes. For other DASD types, the block size should be the largest value (less than or equal to 32760) that best uses the space available on each track. The record length must be 4 bytes less than the block size value.

The SNA 3270 Reformatter Utility uses SYSPRINT to write reports. Normally this is defined to SYSOUT, although you can use a data set instead. If you use a data set, it must have an FBA (fixed block) record format, a record length of 133 and a blocksize that is a multiple of 133.

Running the SNA 3270 Reformatter Utility

The SNA 3270 Reformatter Utility can be run as a batch job or from a CLIST on MVS, or from the WSim/ISPF Interface. You need a region size of at least 275K to run the utility. These storage requirements increase as the number of sessions processed increases.

Note: The NPMLOG data set can be either an FNMVLOG data set from NPM or a GTF trace data set containing a VTAM full buffer trace.

Running the SNA 3270 Reformatter Utility as a batch job

The following is an example job to run the SNA 3270 Reformatter Utility.

```
//LU2RF    JOB //ITPLU2RF EXEC PGM=ITPLU2RF
//STEPLIB  DD DSN=WSIM.SITPLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//NPMLOG   DD DSN=NPM.FNMVLOG.COPY,DISP=SHR
//WSIMLOG  DD DSN=WSIM.ITPLU2RF.WSIMLOG,DISP=OLD
```

Running the SNA 3270 Reformatter Utility from an MVS CLIST

The following is an example CLIST that invokes the SNA 3270 Reformatter Utility.

```
ALLOC DDNAME(SYSPRINT) SYSOUT(A)
ALLOC DDNAME(NPMLOG)  DA('NPM.FNMVLOG.COPY') SHR REUS
ALLOC DDNAME(WSIMLOG) DA('WSIM.ITPLU2RF.WSIMLOG') OLD REUS
CALL 'WSIM.SITPLOAD(ITPLU2RF)'
FREE  DDNAME(SYSPRINT NPMLOG WSIMLOG)
```

Running the SNA 3270 Reformatter Utility from the WSim/ISPF Interface

To invoke the SNA 3270 Reformatter Utility from the WSim/ISPF Interface, follow these steps:

1. Invoke the WSim/ISPF Interface Main panel from ISPF. The method you use to do this depends on how the WSim/ISPF Interface application is installed at your site. If you are not sure how to do this, see your system programmer for assistance.
2. Select option 5 from the WSim/ISPF Interface Main panel and press **Enter**. The Generate Message Decks, STL Programs, and WSim Logs panel is displayed.
3. Select option 3 from this panel and press **Enter**. The Generate WSim Logs from Captured Trace Data panel is displayed.

   Note: You can also type "LU2RF" on the WSim/ISPF Interface Main panel command line and press **Enter** to display this panel.
4. Fill in the appropriate fields on this panel and press **Enter**.

For more information on the WSim/ISPF Interface, refer to Part 1, "General utilities," on page 1.

Specifying execution parameters

There is one execution parameter for the SNA 3270 Reformatter Utility: the print line count. This is the number of lines per report page. The syntax is **PRTLNCNT=***xxx*, where *xxx* is an integer between 35 and 255. If PRTLNCNT is not specified, the default is 60 lines per page.

Understanding SNA 3270 Reformatter Utility output

The SNA 3270 Reformatter Utility writes reports to SYSPRINT: Summary report, Eligible Terminal report and Ineligible Terminal report. There is also a return code, which is described in *WSim Messages and Codes*.

Understanding the Summary Report

This report summarizes the SNA 3270 Reformatter Utility run, showing counts and time stamps for the log processed. An example of this report is shown below:

```
                                         SUMMARY REPORT



 ITPLU2RF RETURN CODE:         0

 LINES PER PAGE      :        60

 LOG RECORDS READ    :       152
 LOG RECORDS SELECTED:       126
 LOG RECORDS IGNORED :        26

 REFORMAT WARNINGS   :         0

 FIRST LOG RECORD    :   7.40.00  FEBRUARY 13, 2002
 LAST LOG RECORD     :   7.42.48  FEBRUARY 13, 2002

 ELIGIBLE SESSIONS   :         6
 INELIGIBLE SESSIONS :         0
```

ITPLU2RF RETURN CODE
: The return code from the SNA 3270 Reformatter Utility. See *WSim Messages and Codes* for a description of each return code.

LINES PER PAGE
: The number of lines per report page on SYSPRINT. The default value is 60, unless altered by the PRTLNCNT execution parameter.

LOG RECORDS READ
: The number of records read from the log.

LOG RECORDS SELECTED
: The number of records selected for processing.

LOG RECORDS IGNORED
: The number of records not selected for processing. Ignored records include:
  - Control records
  - Records for sessions between applications and VTAM.

REFORMAT WARNINGS
: The number of reformat warning messages written to the WSim log data set.

FIRST LOG RECORD
: The time of the first valid record on the log.

LAST LOG RECORD
: The time of the last valid record on the log.

ELIGIBLE SESSIONS
: The number of eligible sessions reformatted by the SNA 3270 Reformatter Utility. The details of these sessions are shown in the Eligible Terminal Report.

INELIGIBLE SESSIONS
: The number of sessions not reformatted by the SNA 3270 Reformatter Utility. The details of these sessions and the reasons why they were rejected are shown in the Ineligible Terminal Report.

Understanding the Eligible Terminal Report

The Eligible Terminal Report has one record for each of the eligible sessions processed by the SNA 3270 Reformatter Utility. An example record is shown below. This record identifies the session and shows time stamps when various events occurred. There are also statistics on the amount of data transmitted and received between the terminal and application. These statistics are only for user data, namely, the FM data request RUs. The TH, RH, and RU types other than FM data are not included in these counts.

```
                                           ELIGIBLE TERMINAL REPORT

 THE FOLLOWING SESSIONS HAVE BEEN REFORMATTED:


 TERMINAL           : TERM1                APPLICATION : PCICS01             LOGMODE :     L3278M2
 FIRST RECORD       :   7.40.00  FEBRUARY 13, 2002
 INIT-SELF TIME     : ITPLU2RF DEFAULT USED
 BIND TIME          :   7.40.02  FEBRUARY 13, 2002
 SDT TIME           :   7.40.02  FEBRUARY 13, 2002
 UNBIND TIME        : NONE FOUND
 LAST RECORD        :   7.42.45  FEBRUARY 13, 2002
 XMIT RECORD COUNT  :        14                    RECV RECORD COUNT   :        16
      MINIMUM LENGTH :        1                         MINIMUM LENGTH :         2
      MAXIMUM LENGTH :       24                         MAXIMUM LENGTH :       797
      AVERAGE LENGTH :        7                         AVERAGE LENGTH :       258
      TOTAL LENGTH   :       93                         TOTAL LENGTH   :     4,135
 TOTAL XMIT + RECV  :     4,228                    AVERAGE XMIT + RECV :       141
```

TERMINAL
The 3270 display terminal name (the secondary LU2).

APPLICATION
The application name (the primary LU). If the application sends an "UNBIND with BIND forthcoming", this field shows the original application name followed by the application name from the last BIND for this session.

LOGMODE
The logmode used for this session.

FIRST RECORD
The time stamp of the first record for this session on the log, usually the time of the CINIT.

INIT-SELF TIME
If an INIT-SELF is on the log, then INIT-SELF TIME contains its time stamp. If not, the SNA 3270 Reformatter Utility creates a default INIT-SELF using information derived from the CINIT for this session, and the message "ITPLU2RF DEFAULT USED" appears instead of the time stamp.

BIND TIME
The time stamp of the BIND record.

SDT TIME
Time stamp of the SDT record.

UNBIND TIME
The time stamp of the UNBIND record, or "AFTER TRACE ENDED" message if the UNBIND is not present on the log.

LAST RECORD
The time stamp of the last record processed for this session.

RECORD COUNT
The number of FM data request RUs found for this session. (Transmit and receive values shown separately.)

MINIMUM LENGTH
The size in bytes of the smallest FM data request RU for this session. (Transmit and receive values shown separately.)

MAXIMUM LENGTH
The size in bytes of the largest FM data request RU for this session. (Transmit and receive values shown separately.)

AVERAGE LENGTH
The average size in bytes of all FM data request RUs for this session. (Transmit and receive values shown separately.)

TOTAL LENGTH
The sum of all FM data request RU lengths for this session. (Transmit and receive values shown separately.)

TOTAL XMIT+RECV
The sum of all FM data request RU lengths for both transmit and receive RUs.

AVERAGE XMIT+RECV
The average size of all FM data request RUs transmitted and received by this session.

A note appears at the end of the statistics for a particular session if one of the following conditions exist:

- The session is LU type 0 and the BIND specifies that the FM profile is type 2 and the TS profile is type 2. The note indicates that a 3270 data stream is assumed.
- Extraneous records are found before the BIND. The note indicates that extraneous records are ignored.

If no eligible sessions were found, then the following message appears instead of the report shown above.

```
NO ELIGIBLE SESSIONS FOUND.
```

Understanding the Ineligible Terminal Report

If all sessions were successfully processed, then the following message appears.

```
NO INELIGIBLE SESSIONS FOUND.
```

However, if there are any ineligible sessions, the Ineligible Terminal report uses the same format as the Eligible Terminal Report with the addition of a reject reason line. This reject reason can have one of three values:

1. `NO BIND RECORD FOUND FOR THIS SESSION.`

   If a BIND record is not on the NPM log, then no WSim display (DSPY) records for this session can be created. Without these DSPY records, the WSim Log Script Generator (ITPLSGEN) cannot create scripts for this session.
2. `THIS SESSION IS NOT LU TYPE 2 OR 3270 LU TYPE 0.`

   The SNA 3270 Reformatter Utility only processes records between terminals and applications. If the BIND specifies that the session is not LU type 2 or 3270 LU type 0, the session is not reformatted.
3. `ITPLU2RF WARNING MESSAGES WRITTEN TO LOG.`

   You can find the text of any warning messages by running the loglist utility for this session. This is described in “Using the WSim log data set.” The SNA 3270 Reformatter Utility continues processing this session; however there will be missing WSim display (DSPY) records in the WSim log data set. You may not be able to use other WSim utilities such as the Log Script Generator (ITPLSGEN).

Using the WSim log data set

The WSim log data set created by the SNA 3270 Reformatter Utility consists of four types of records:

Data records
: Standard WSim log data set records (XMIT and RECV) containing the PIU data selected from the log.

Display records
: Standard WSim display (DSPY) records containing screen images derived from the PIU data on the log.

Warning records
: Log (LOG) records containing messages about reformatting errors. These messages are documented in *WSim Messages and Codes*.

Comments for ITPLSGEN
: Log (LOG) records containing device characteristics information for this

terminal. These records are used by the Log Script Generator (ITPLSGEN) to add IDC-like prologue comments in STL programs and WSim message decks.

Formatting the WSim log data set

Each record type can be printed by the Loglist Utility (ITPLL). For example, the following loglist commands print all data, display, warnings, and comments for ITPLSGEN for terminal TERM1:

```
TERM TERM1
DATA
DSPLY ATTR
LOG
RUN
END
```

Search for "ITP180" to find SNA 3270 Reformatter Utility warning records. This locates all ITP180n messages, which are detailed in *WSim Messages and Codes*.

Generating a WSim script

The SNA 3270 Reformatter Utility is specifically designed to create WSim log data sets for the Log Script Generator (ITPLSGEN). See Chapter 16, "Generating scripts from IDC or WSim log data sets," on page 213 for details on how to use ITPLSGEN.

The SNA 3270 Reformatter Utility creates device characteristics comment records in the STL programs or message decks. These comments describe the device characteristics for this terminal. The defaults used by the SNA 3270 Reformatter Utility are shown below.

```
ALTCSET=APL              APLCSID=(963,310)
BASECSID=(697,37)        CCSIZE=(9,16)     COLOR=MULTI
DBCS=NO
DISPLAY=(24,80,32,80)    DLOGMOD=xxxxxxx   EXTFUN=YES
FLDOUTLN=NO              FLDVALID=NO       HIGHLITE=YES
MAXNOPTN=0               PS=(S,S,T,T,S,T)  UOM=INCH
```

Note that the DISPLAY values may be altered by the BIND.

The SNA 3270 Reformatter Utility also creates comment records whenever a 3270 query reply log record is processed that indicates a new value for any of the device characteristics. If all of the values in the query reply are the same as the default values, no comment records are created when the query reply is processed. The values held on the 3270 query reply are combined with the following defaults to create a new set of device characteristics:

```
ALTCSET=NONE
BASECSID=(697,37)        CCSIZE=(9,16)     COLOR=GREEN
DBCS=NO
DISPLAY=(24,80,24,80)    DLOGMOD=xxxxxx    EXTFUN=NO
FLDOUTLN=NO              FLDVALID=NO       HIGHLITE=NO
MAXNOPTN=0               PS=NONE           UOM=INCH
```

Calculating response times

The response times logged during session data collection can be printed running the response time utility for the WSim log data set created by the SNA 3270 Reformatter Utility. The response time utility is described in Part 1, "General utilities," on page 1.

Comparing WSim log data sets

The Log Compare Utility (ITPCOMP) can process WSim log data sets created by the SNA 3270 Reformatter Utility. Use the reformatted WSim log data set as the MASTER compare log. The TEST log should be the WSim log data set created when you ran the scripts generated by ITPLSGEN from the reformatted log. Refer to Part 1, “General utilities,” on page 1.

Understanding SNA 3270 Reformatter Utility restrictions

You need NPM Version 1 Release 4 or later to capture logs for reformatting by the SNA 3270 Reformatter Utility. VTAM PIU logs captured by earlier releases of NPM may not reformat correctly. If the SNA 3270 Reformatter Utility fails to process an NPM log, then a non-zero return code is shown in the Summary Report (as described in “Understanding the Summary Report” on page 223). This return code is described in *WSim Messages and Codes* and explains why the problem has occurred. If you are still not clear about the cause of the error, here are some ideas that may help:

1. The SNA 3270 Reformatter Utility has many of the same restrictions as IDC. The SNA 3270 Reformatter Utility supports the following 3270 functions:
   - Color
   - Highlighting
   - Character sets for base and APL
   - Field validation
   - Alphanumeric partitions
   - Magnetic Stripe Reader
   - Selector Light Pen
   - PIU segmentation
   - PIU chaining
   - AID validation
   - Local Clears
   - Erase-EOF
   - ERIN.
2. Both IDC and the SNA 3270 Reformatter Utility do not support the following functions:
   - File transfer
   - All Points Addressable (APA) graphics
   - Auxiliary devices.
3. The SNA 3270 Reformatter Utility can only process LU type 2 and 3270 LU type 0 sessions with BINDs. If a session is not one of these LU types, no records are reformatted for this session. If the session is one of these LU types but does not have a BIND, then the session is not reformatted. The SNA 3270 Reformatter Utility flushes any data for a session of unknown LU type. If a BIND is later found that defines the session to be either LU type 2 or 3270 LU type 0, the remainder of the session records are processed.
4. If the SNA 3270 Reformatter Utility does not recognize your log as having the correct format, check that the options used for data collection were specified correctly.
5. If you are tracing an application, this does not mean you can create an STL program or WSim message deck to simulate the application. The Log Script

Generator will only create scripts for the secondary LU (a 3270 terminal) and not for the primary LU (the application).

6. If the Log Script Generator (ITPLSGEN) fails to create a script from the WSim log data set produced by the SNA 3270 Reformatter Utility, make sure the SNA 3270 Reformatter Utility completed with a return code of 0 or 4. If the return code is 8, check for warning messages reported for this session. See "Understanding the Ineligible Terminal Report" on page 226 for details.
7. The NPM FNMVLOG or VTAM full buffer trace data set contains PIU data and not screen images. The SNA 3270 Reformatter Utility derives the screen images from the PIU data only. This means that local 3270 functions such as Erase-EOF and Erase Input (which do not appear in the PIU) will be evaluated from the state of all fields in the VTAM PIU data. The display (DSPY) records in the WSim log data set may not always reflect exactly the screen images seen at session data collection time.

   The following are examples of when this might happen:

   - When nulls are imbedded between significant data within a field on the screen image. The nulls are suppressed in the PIU and in the resulting display records in the WSim log data set.
   - When a short AID (such as PA1) is entered. There may be data on the screen image that does not appear in the resulting display records, since it is not included in the PIU.
   - When a local clear is done at the terminal. Typically, it is impossible to determine from the PIU data that a local clear occurred. So the resulting display records may reflect the screen image before the clear request.
   - When a trigger attribute is present on the screen and a trigger action occurs. The final cursor position may be incorrect, since the trigger action cursor movement is not reflected in the data stream.

SNA 3270 Reformatter Utility return codes

SNA 3270 Reformatter Utility issues the following return codes:

Code	**Meaning**
0	SNA 3270 Reformatter Utility has completed without any errors, warning messages or invalid sessions.
4	SNA 3270 Reformatter Utility has completed without any errors but has detected at least one invalid session on the input log, probably because of a missing BIND record for that session. Details of invalid sessions can be found in the ineligible terminal report.
8	SNA 3270 Reformatter Utility has completed with a least one warning message written to the WSim log. Any sessions with warning messages can be found in the ineligible terminal report.
12	An unrecoverable I/O error occurred when trying to read from the NPMLOG DD. Make sure the NPMLOG DD is allocated to the correct data set.
16	The log has an unrecognized data format and is probably not a log at all. Make sure the NPMLOG DD is allocated to the correct data set.
20	Unable to open the log. Check the allocation of the NPMLOG DD.
24	Lost data on the log. The NPM log has a record with an event identifier of X'E107', indicating that data has been lost on the log. The log is invalid and must be recreated.
28	Unable to open the SYSPRINT DD. Check the allocation of the SYSPRINT DD.
32	Unable to open the WSIMLOG DD. Check the allocation of the WSIMLOG DD.

36	Insufficient storage to run ITPLU2RF. Allocate more storage to ITPLU2RF before rerunning the job.
40	PRTLNCNT parameter incorrect. Make sure the execution parameter has the correct format.
44	An unrecoverable I/O error occurred when trying to write to the WSIMLOG DD. Make sure the WSIMLOG DD is allocated correctly and has sufficient space in the dataset and volume.
48	The log is empty. Make sure the NPMLOG DD is correctly allocated.

Chapter 18. Using the Script Generator Utility

The Script Generator Utility (ITPSGEN) enables you to generate WSim message generation decks for the terminals in the network definition. Decks are generated using network definitions and captured data traffic from live runs of the system under test. You can specify either existing or new network definitions as input.

You must put the data captured for the Script Generator Utility in a specified format and sort it by resource name, date, and time. A program provided as part of the Script Generator Utility (ITPVTBRF) accepts other formats of captured data and reformats the data for input to ITPSGEN, the utility program that actually generates the message generation decks.

You must follow five steps to generate message generation decks using this script generator:

1. Obtain a trace of system activity.
2. Reformat the trace output if it is not in the format required for the script generator utility.
3. Sort the reformatted output using any standard sort program.
4. Define the network.
5. Generate the message generation decks.

Figure 45 on page 232 lists the programs you use for each step. The following sections discuss the programs provided, give instructions for their use, and provide general information regarding the script generation process.

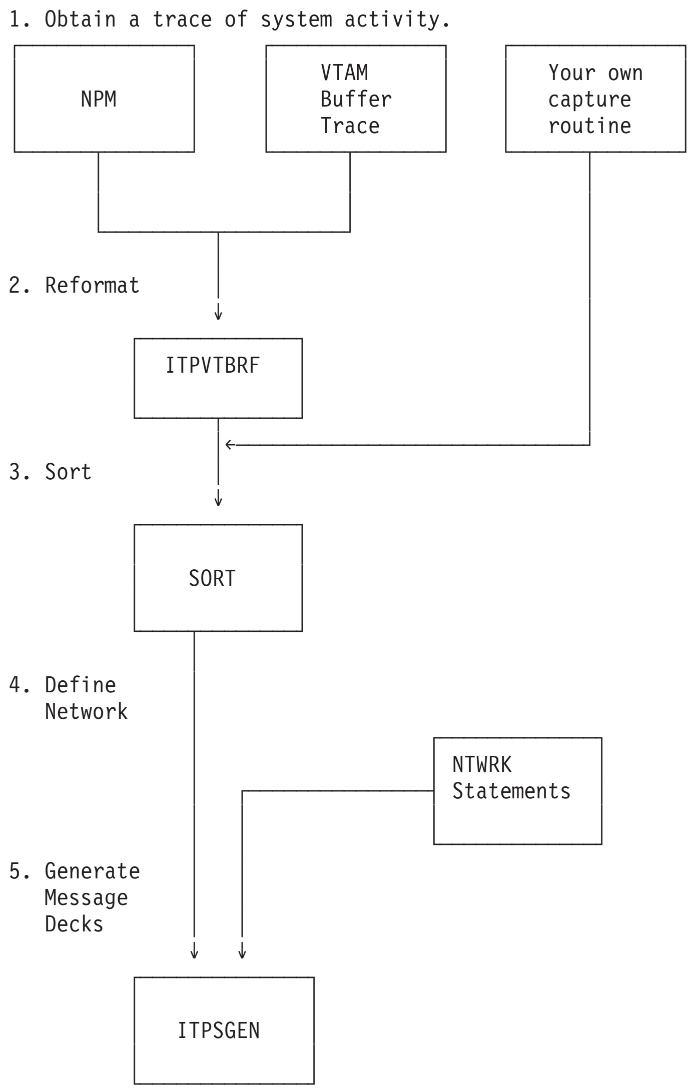


Figure 45. Generating scripts with the Script Generator Utility

Operational suggestions

ITPSGEN builds message generation decks to re-create the captured terminal data traffic. The quality of the resulting scripts cannot be better than the actual data traffic captured. The following paragraphs describe two methods that you can use to capture terminal data traffic, some of the problems you may encounter, and possible solutions to the problems.

Capture single terminal traffic

You can use the following method in situations where the WSim scripts must generate a particular set of host application transactions. This method has the advantage of being easily controlled.

1. Identify a single terminal and operator to initiate a conversation with the host application, execute a predefined set of transactions, and end the conversation with the host application.
2. Start the capture routine.

3. Have the operator execute the conversation with the host application using the identified terminal.
4. Stop the capture routine.
5. Reformat and sort the captured terminal data traffic.
6. Create the network definition for your simulation.
7. Run ITPSGEN to produce a message generation deck.
8. Modify the network definition to force each terminal to execute the generated message generation deck using the PATH operand.

   Note: You may also want to modify the generated deck to include references to user tables (UTBL), counters, or random numbers. If you do not do this, all the simulated terminals will send the same message traffic as the terminal that was traced. This can cause problems because all simulated terminals will try to access the same data base records.
9. Start WSim. Initialize and start the network created by the previous step. The WSim network, through the generated message generation deck, will duplicate the captured host application conversation for each simulated terminal defined in the network.

Capture system terminal traffic

The following method will capture all the terminal data traffic within the system. This method can be used to recreate the terminal-to-host application conversations that are unique to each terminal. The advantage of this method is that the captured data represents the actual terminal data traffic occurring during the capture time frame. The disadvantages of this method are that the state of each terminal is not under your complete control and the transactions entered by the terminal operators may not be the type required for the host application test.

1. Start the capture routine.
2. Start the host application. This should be performed during a time when a heavy load can be expected.
3. Allow the terminal data traffic to be captured for a period of time.
4. Stop the capture routine.
5. Reformat and sort the captured data.
6. Create the network definition for your simulation.
7. Run ITPSGEN to produce the message generation decks for each terminal.
8. Start WSim. Initialize and start the network created by the previous step. The WSim network will duplicate the terminal to host application conversations previously captured.

Step 1. Obtaining a trace of system activity

Before you can use the Script Generator Utility to produce message generation decks, you must obtain a trace of system activity. Several possible methods of obtaining a trace are discussed here, but there are other methods as well. Any method that produces data in the required format is acceptable.

NPM VTAM Log

The NetView* Performance Monitor (NPM) provides a facility for capturing the path information units (PIUs) for selected logical units. The data is collected in a VTAM log data set. If you use this facility, be sure to specify VLOG=1 and

MIN=NO on the NPM Start Session Collection Menu. For more information about using the NPM data collection facility, refer to the *NetView Performance Monitor User's Guide and Reference*.

Reformatting

You can use the WSim reformatter, ITPVTBRF, to reformat the VTAM log data set. See "ITPVTBRF" on page 235 for more information.

VTAM buffer trace

You can use the VTAM buffer trace together with the Generalized Trace Facility (GTF) to obtain a trace of system activity.

You should start GTF with the USR option (TRACE=USR).

Limitations

Data is truncated if it is longer than one GTF trace record. This limitation may not be a problem, however, if the terminals communicating with VTAM applications do not send long messages. Also, data sent to the terminals is usually only used to determine delays and values for RESP data, so it normally does not matter if it is truncated.

The normal VTAM buffer trace and the GTF facility can record a maximum (excluding SNA headers) of 195 bytes. With VTAM Version 4 Release 1 and later, you can use the full buffer trace option. When you use this option, VTAM records the entire PIU and ITPVTBRF can process up to 8000 bytes per PIU.

Reformatting

You can use the WSim reformat program, ITPVTBRF, to convert the GTF data set to the format needed by ITPSGEN. This program is described in "ITPVTBRF" on page 235. You can find additional information and instructions for using the GTF facility in conjunction with VTAM buffer trace in *ACF/VTAM Operation* and in the appropriate service aids manual for the system you are using.

The VTAM buffer trace is for MVS only.

Your own capture routine

You can also write your own capture routine to obtain input to ITPSGEN. You should block the input data set for ITPSGEN with variable length records (RECFM=VB) and a BLKSIZE of up to 32760 bytes. You must sort the tape or data set by terminal name, date, and time before using it as input to ITPSGEN. The format of each record is described below.

Offset		**Length**	
Decimal	**Hex**	**(in bytes)**	**Description**
0	(0)	2	Logical record length (in binary)
2	(2)	2	Reserved - always zero
4	(4)	4	Time of day in binary hundredths of seconds
8	(8)	4	Julian date in packed decimal form, *yyyydddC*, where *yyyy* is the year and *ddd* is the day
12	(C)	8	Resource name
20	(14)	2	Reserved
22	(16)	2	Session number
24	(18)	1	Transmission Header (TH) byte containing Segmenting and Expedited Flow Indicators
25	(19)	3	Request/Response Header (RH)
28	(1C)	1	Flags
			1... .... Transmit record from resource

```
                                .1.. ....  Record to/from SSCP
                                ..1. ....  End of block
                                ...1 ....  Subblock
                                .... 1...  Primary LU
  29     (1D)       1       Format identifier (FID)
  30     (1E)       2       Data length
  32     (20)   variable    Data - maximum 6000 bytes
```

Where to take the trace

There are two different situations for which you may want to take traces. The first situation is where you want to generate scripts for simulating terminals. Here you have the standard terminal-to-application sessions. The other situation is where you want to generate scripts for simulating applications. This is normally in a LU6.2 environment, where you have applications in session with other applications.

When you generate scripts for terminals, you can trace either the terminal or the application. The trace records look the same whichever is traced. Tracing the resources to be simulated in that case may cut down the size of the trace if other terminals also have sessions with that application, but otherwise there is no difference.

When both applications are associated with the same VTAM, you must trace the application to be tested rather than the application to be simulated. When both applications are associated with different VTAMs, you must take the trace in the VTAM associated with the application to be tested and not that of the application to be simulated.

Step 2. Reformatting the trace output

If the trace of system activity obtained is not in the format that the Script Generator Utility requires, the data must be reformatted before you can use it for generating message generation decks. WSim provides a program (ITPVTBRF) to help you reformat your data.

ITPVTBRF

ITPVTBRF reformats VTAM buffer trace records recorded using the GTF or the NPM data collection facility. ITPVTBRF produces reformatted records for the originating node name (for data input to VTAM) and for the destination node name (for data output from VTAM). If the opposite node name is VTAM, the to/from SSCP flag is turned on in the reformatted record. ITPVTBRF uses one of two inputs, the GTF trace data set or the VTAM log data set created by NPM, and produces one output, a reformatted trace data set.

Note: ITPVTBRF does not reformat the VTAM buffer trace records produced on VM, but NPM records produced on VM can be used. Also, ITPVTBRF only recognizes PIUs using extended addresses.

If ITPVTBRF detects LU type 6.2 BINDs in the VTAM buffer trace data, it assigns a session number to each reformatted trace record, matching addresses in the traced data to addresses in the individual BINDs. This enables the Script Generator Utility to generate separate MSGTXT decks for each LU type 6.2 session.

ITPVTBRF sets a special return code if a record produced for an originating node contains truncated data. In this case, you may want to examine the VTAM buffer trace using the IPCS or ACF/TAP and inspect the truncated data. Records produced for destination node names (data received by the terminal) are not checked for truncated data.

ITPVTBRF can process a trace of cross-domain sessions as well as terminal session data. It looks at cross-domain commands to determine session partners and addresses.

Note: ITPVTBRF must have information from the first volume of a multiple-volume NPM log data set before it can reformat your data. If ITPVTBRF does not reference this volume before other volumes in the log data set, the program sets a return code of 8 (RC=8) indicating that the data on the input data set was formatted incorrectly.

Running ITPVTBRF from the WSim/ISPF Interface

To run ITPVTBRF from the WSim/ISPF Interface, follow these steps:

1. Invoke the WSim/ISPF Interface main panel from ISPF. The method you use to do this depends on how the WSim/ISPF Interface application is installed at your site. If you are not sure how to do this, see your system programmer for assistance.
2. Select option 5 from the WSim/ISPF Interface main panel and press **Enter**. The Generate Message Decks and STL Programs panel is displayed.
3. Select option 2 from this panel and press **Enter**. The Generate Message Decks from Captured Trace Data panel is displayed.
4. Select option 1 from this panel and press **Enter**. The Reformat NPM and VTAM Buffer Traces panel is displayed.

   Note: You can also type “NPMVTAM” on the WSim/ISPF Interface main panel command line and press **Enter** to display this panel.
5. Fill in the appropriate fields on this panel and press **Enter** to run ITPVTBRF.

For more information on the WSim/ISPF Interface, refer to Part 1, “General utilities,” on page 1.

Using JCL to run ITPVTBRF

The JCL shown in the example below executes ITPVTBRF. The TAPEIN DD statement specifies the input data set, and the TAPEOUT DD statement specifies the output data set.

```
//VTBRFJOB JOB
//JOBLIB   DD   DSN=WSIM.SITPLOAD,DISP=SHR
//VTBRF    EXEC PGM=ITPVTBRF
//TAPEIN   DD   UNIT=TAPE,DISP=OLD,VOL=SER=TAPEIN,LABEL=(,NL)
//TAPEOUT  DD   UNIT=TAPE,DSN=TAPEOUT,DISP=(NEW,KEEP),LABEL=(,NL)
```

New ITPVTBRF execution parameters

You can use the following execution parameters when running ITPVTBRF:

INVERT
: Allows scripts to be generated in situations where the VTAM buffer trace was active for an LU that did not capture "inbound" data to VTAM. When INVERT is specified, "outbound" data from VTAM will be reformatted to allow for script generation by ITPSGEN.

SSCPNAME=*sscp_name*
: Allows an SSCP name other than the default ("VTAM") to be specified.

ITPVTBRF return codes

At the end of execution, ITPVTBRF sets a return code to indicate the status of the execution. ITPVTBRF execution ends prematurely for all return codes except 0, 28, 32, and 40.

Code	Meaning
0	Execution ended successfully.
4	An immediate end of file was encountered on the input data set.
8	The data on the input data set has the incorrect format.
12	An unrecoverable I/O error was encountered on TAPEIN.
16	An unrecoverable I/O error was encountered on TAPEOUT.
20	The TAPEIN data set could not be opened.
24	The TAPEOUT data set could not be opened.
28	No data was written to TAPEOUT data set.
32	At least one record with truncated data was processed.
40	A lost data record was encountered on the NPM VTAM log data set.

Step 3. Sorting the trace data

The Script Generator Utility assumes that the trace data has been sorted into ascending order based on the name, date, and time fields. You can sort the data with any standard sort program.

If you use the DFSORT program product to sort a data set that is already formatted correctly for input to ITPSGEN, you can code the SORT control statement as follows:

```
SORT  FIELDS=(13,8,CH,A,23,2,FI,A,9,4,PD,A,5,4,FI,A),EQUALS
```

In the above example, the data set is sorted by LU name (13,8,CH,A), then by session number (23,2,FI,A), then by date (9,4,PD,A), and finally by time (5,4,FI,A).

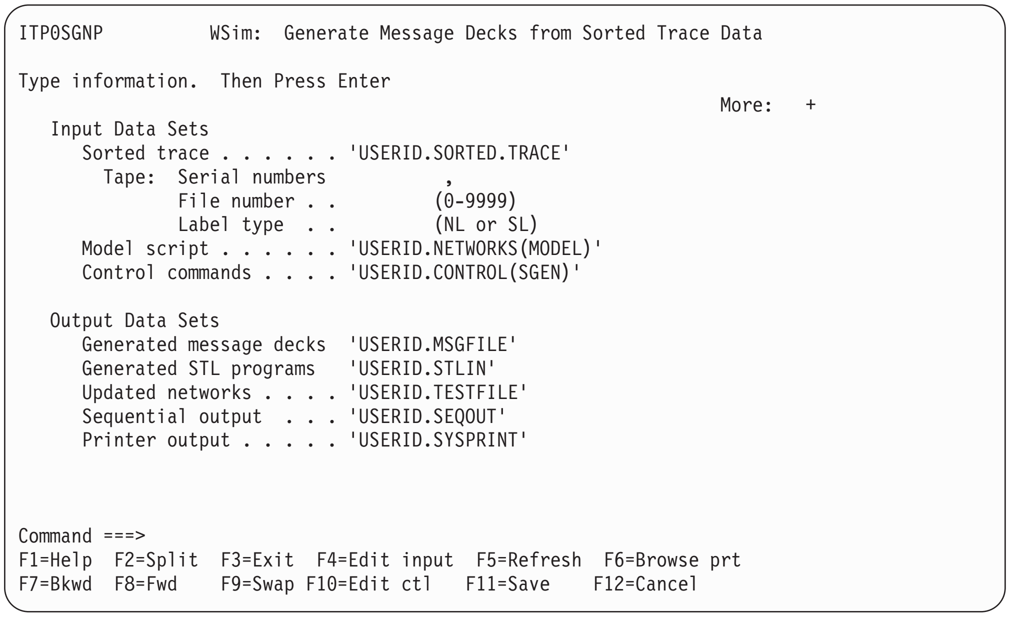
Be sure to specify the sort control information so that the sorted output data set has the same block size as the reformatted trace data input. The default block size is 8192 but block sizes up to a maximum of 32760 can be specified.

You can also sort the trace data using the WSim/ISPF Interface. To do this, follow these steps:

1. Invoke the WSim/ISPF Interface main panel from ISPF. The method you use to do this depends on how the WSim/ISPF Interface application is installed at your site. If you are not sure how to do this, see your system programmer for assistance.
2. Select option 5 from the WSim/ISPF Interface main panel and press **Enter**. The Generate Message Decks and STL Programs panel is displayed.
3. Select option 2 from this panel and press **Enter**. The Generate Message Decks from Captured Trace Data panel is displayed.
4. Select option 2 from this panel and press **Enter**. The Sort Trace Data Using DFSORT panel is displayed.

   Note: You can also type “SORTTRCE” on the WSim/ISPF Interface main panel command line and press **Enter** to display this panel.

When generating STL scripts using the WSim/ISPF Interface, you must specify a data set to contain the generated STL source. Use the field labeled **Generated STL programs** on panel ITP0SGNP to specify this data set. A member will be created in this data set for each STL MSGTXT that is generated. The data set can have the same attributes as those used for the generated message decks data set. The following is an example of the ITP0SGNP panel as it might look when generating STL scripts.



```
ITP0SGNP         WSim:  Generate Message Decks from Sorted Trace Data

Type information.  Then Press Enter
                                                                 More:   +
   Input Data Sets
      Sorted trace . . . . . . 'USERID.SORTED.TRACE'
        Tape:  Serial numbers            ,
               File number . .         (0-9999)
               Label type  . .         (NL or SL)
      Model script . . . . . . 'USERID.NETWORKS(MODEL)'
      Control commands . . . . 'USERID.CONTROL(SGEN)'

   Output Data Sets
      Generated message decks  'USERID.MSGFILE'
      Generated STL programs   'USERID.STLIN'
      Updated networks . . . . 'USERID.TESTFILE'
      Sequential output  . . . 'USERID.SEQOUT'
      Printer output . . . . . 'USERID.SYSPRINT'



Command ===>
F1=Help  F2=Split  F3=Exit  F4=Edit input  F5=Refresh  F6=Browse prt
F7=Bkwd  F8=Fwd    F9=Swap F10=Edit ctl   F11=Save    F12=Cancel
```

Figure 46. Generating message decks from sorted trace data

5. Fill in the appropriate fields on this panel and press **Enter**.

For more information on the WSim/ISPF Interface, refer to Part 1, "General utilities," on page 1. Refer to the *DFSORT Application Programming Guide* for additional information on sorting.

Step 4. Defining the network

This section describes the network definitions that are used as input to the Script Generator Utility.

The network definitions used as input to ITPSGEN should be complete, syntactically correct WSim networks. They must be capable of being processed without errors by the WSim Preprocessor. They do not need to be preprocessed before being used as input.

Each network should have a single message generation sequence PATH statement defined (path 0), with a PATH=(0) operand specified at the network level. You should not include any other message generation sequence PATH operands or statements. If you select the network update option, PATH statements will be added to the network. The PATH named on the network statement will be the PATH used for any terminals or devices for which no messages are found on the trace data set; thus, the path statement should name a deck that contains only a WAIT statement followed by a BRANCH back to that WAIT statement.

ITPSGEN uses the WSim network definitions to determine the terminal names for which decks are to be generated. These names, taken from the LU statements in the definitions, must correspond to the resource names used in the trace data set. There should be no duplication of names for terminals defined in the input network definitions.

The terminal type for each name and the think time option (immediate or unlock) are determined from the network definition.

A MAXSESS operand is added to all applicable LU types when the network update option is specified.

A terminal name will be considered eligible for message generation if it is a type supported by ITPSGEN and if it does not have a PATH operand or if a PATH=(0) operand is specified or defaulted. If the PATH=(0) operand is specified on the LU statement, a duplicate operand will be generated if the network update option is used.

The user time interval (UTI) defined for a network or for an individual terminal is also taken from the network definitions. The UTI is used in calculating delays.

Below provides an example network definition for ITPSGEN.

```
SGENNET  NTWRK BUFSIZE=2048,         BUFFER SIZE FOR MESSAGES
               DELAY=A1,             RANDOM AVERAGE DELAY FROM 0->2 TIMES 1
               DISPLAY=(24,80,32,80), PRIMARY & ALTERNATE DISPLAY SIZES
               DLOGMOD=D4A32783,     LOGON MODE TABLE ENTRY FOR LOCAL LU2
               EXTFUN=YES,           3270 EXTENDED FUNCTION SUPPORT PROVIDED
               HEAD='SCRIPT GENERATOR', HEADER FOR ONLINE REPORTS
               INIT=SEC,             WSim WILL INITIATE SECONDARY SESSIONS
               ITIME=1,              PRINT NTWRK INTERVAL REPORT EACH MINUTE
               LUTYPE=LU2,           TYPE OF SIMULATED SNA LOGICAL UNIT
               MAXSESS=(0,1),        ONLY ONE SECONDARY SESSION WITH LU2
               OPTIONS=(DEBUG,CONRATE,MONCMND), SNA SEGM/RU'S & OPCMD
               PATH=(0),             DEFAULT PATH STATEMENT
               RESOURCE=WSIMPLU,     NAME OF APPLICATION UNDER TEST
               RSTATS=YES,           ONLINE RESPONSE TIME STATISTICS
               THKTIME=UNLOCK,       START DELAY CALC AFTER KEYBOARD RESTORE
               USERAREA=4,           SIZE OF STORAGE FOR SCRATCH PAD AREA
               UTI=100               1 SEC. USER TIME INTERVAL FOR DELAYS
*


0        PATH  SGENWAIT
*
         VTAMAPPL APPLID=WSIM0101,   APPLID DEFINED TO VTAM
               BUFSIZE=2048          BUFFER SIZE OF SIMULATED LU
*
TAF05F01 LU    LOGDSPLY=BOTH,        LOG SCREEN IMAGES
               MSGTRACE=YES          LOG MESSAGE GENERATION TRACE RECORDS
*
SGENWAIT MSGTXT
LOOP     WAIT
         BRANCH LABEL=LOOP
         ENDTXT
```

Step 5. Generating the message generation decks

The Script Generator Utility is provided by WSim to create message generation decks. It can also update the required input network definitions to reflect the message generation decks created.

To run ITPSGEN, you must have a sorted trace data set and at least one WSim network definition. You can also include control commands and additional network definitions.

The principal outputs are the generated message generation decks. Other outputs are reports, updated network definitions, and a composite sequential data set containing all message generation decks and network definitions.

The sorted input data set is scanned for records that contain terminal names in the submitted networks. Message generation decks with the same names are generated based on the captured data. These decks are written to a partitioned data set as they are built. After all the decks are generated, the network definitions can be updated to reflect the new decks. You can also copy all the message generation decks and updated networks to a single sequential data set.

ITPSGEN terminal types supported

The terminal types ITPSGEN supports include most of the types that can be specified on the LU statements in a WSim network. They are:

LU0	LU6	LU1	LU62	LU2
LU4				

You can generate a deck for a terminal type that is different from the terminal type for which the data was captured. You should check the deck carefully because it is not always possible to translate the deck accurately. *ITPSGEN does not support the selector pen for 3270 devices*. The program does support partitions and provides limited support for the trigger field AID.

For 3270 terminals, generated scripts assume all data can be entered by the simulated operator during WSim message generation. Data in preinitialized fields may cause WSim to log informational messages during message generation. You may need to modify your script if this happens.

Running ITPSGEN

You can invoke ITPSGEN from the WSim/ISPF Interface. You can also run ITPSGEN as a standard job on MVS. “Using JCL to run ITPSGEN” on page 252 contains an example of ITPSGEN JCL. “Using a CLIST to run ITPSGEN” on page 253 contains a sample CLIST for running ITPSGEN under TSO.

ITPSGEN execution parameters

You can specify the following parameters for ITPSGEN.

CTL
Specifies that control commands should be read from the CTLIN data set.

NOCTL
Specifies that no control commands should be read from the CTLIN data set and default values for the control commands are used. If neither CTL nor NOCTL is specified, CTL is the default.

PRTLNCNT=*nnn*
Specifies the maximum number of lines printed on an output page before beginning a new page. The *nnn* variable is an integer from 35 to 255. The default value for *nnn* is 60.

ITPSGEN data set requirements

The following data sets are required to run ITPSGEN:

INITDD DD	Defines a partitioned data set to receive the network being processed. The data set can be the same as the MSGDD data set. The BLKSIZE for the data set must be a multiple of 80.
MSGDD DD	Defines a partitioned data set to receive the message generation decks being processed. The data set can be the same as the INITDD data set. The BLKSIZE for this data set must be a multiple of 80.
SYSUT2 DD	Defines a partitioned data set that ITPSGEN will use as work space for storing network definition statements. The data set should be large enough to contain all network definitions being processed.
SYSUT3 DD	Defines a partitioned data set that ITPSGEN will use as work space for storing the message generation decks. The data set should be large enough to contain all message generation decks being processed. It must not be the same data set defined by SYSUT2.
SYSPRINT DD	Defines the output printer or data set to receive the printed output of this job.
MSGTXT DD	Defines the output data set to contain the generated message generation decks. The BLKSIZE for this data set must be a multiple of 80.
TAPEIN DD	Defines the input trace data set. The default block size is 8192 but block sizes up to a maximum of 32760 can be specified.
SYSIN DD	Specifies the data set containing the input data for ITPSGEN (network definitions and associated message generation decks not created by ITPSGEN).
RATEDD DD	Defines the data set that contains the rate table to be used if the network uses a rate table. This data set is optional.
NTWRK DD	Defines the data set to contain the updated network definitions if you request the network update option. The BLKSIZE for this data set must be a multiple of 80. This data set is optional.
SEQOUT DD	Defines the sequential data set to contain all generated message generation decks and updated network definitions if you request the sequential output option. The BLKSIZE for this data set must be a multiple of 80. This data set is optional.
CTLIN DD	Defines the sequential data set containing control commands for ITPSGEN. The BLKSIZE for this data set must be a multiple of 80. This data set is optional.

The relationship between the SYSIN, TAPEIN, INITDD, MSGDD, NTWRK, and MSGTXT data sets is shown in Figure 47.

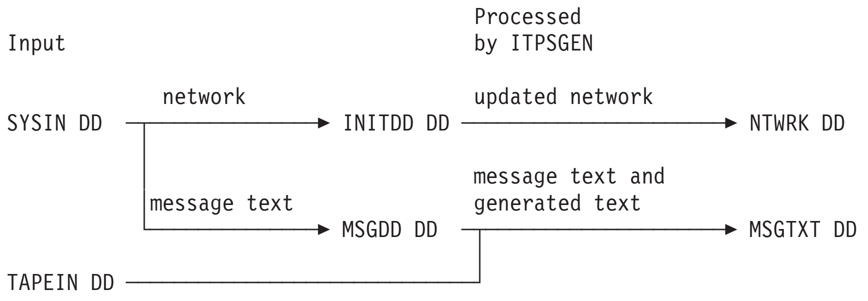


Figure 47. Relationship between ITPSGEN data sets

Note: The MSGTXT, MSGDD, NTWRK, and INITDD DD statements can all reference the same data set, or they can reference different data sets.

ITPSGEN control commands

ITPSGEN control commands are 80-byte records from the data set defined by the CTLIN DD statement. There can be only one command per record and each command must be complete on one record. All data must be contained in columns 1 to 71 of the record. A command can start in any column of the record. Operands must be preceded by at least one blank. Comments can be entered after the operands. Comments must be preceded by at least one blank. A command with an asterisk (*) in the first position is considered a comment.

Understanding control command description conventions

The following conventions are used in the control command descriptions:

- Capital letters represent values you code directly without change.
- Italics represent parameters for which you must supply a value.
- Brackets, [ ], enclose operands or symbols that are either optional or conditional.

  An optional operand is an operand that you may choose to code or omit, independent of other operands. Omitting it may cause a specific default value to be assumed. The default value is always given in the operand description.

  A conditional operand is an operand that you may need to code or omit, depending on how you code (or omit) other operands on the control command. For each conditional operand, the conditions under which you should code or omit the operand are indicated.
- Braces, { }, indicate that an operand has a value that you must choose from the stacked items.
- An ellipsis in parentheses, (...), indicates that you may code a sequence of values within parentheses.
- Default values are underlined.

DELAY and NODELAY commands

```
DELAY [ISTART]
```

This command specifies that DELAY statements will be inserted in the message generation decks between each message. The delays are calculated from the time stamps on the log data set and assume the user time interval (UTI) as specified for that device. The delays are rounded to the next higher multiple of the UTI. If the UTI is 0, no DELAY statements will be inserted for that terminal.

If you specify THKTIME=UNLOCK in the network definition for the terminal, the delay calculated is from the time of the last receive record to the time of the transmit record. If you specify THKTIME=IMMED in the network definition for the terminal, the delay calculated is from the time of the first record of the previous transmit message to the time of the first record of the current message.

The initial delay (before the first message) can be calculated only if a start time is specified on the TIME control command or if a BIND or SDT command has been received and recorded before the first message. If ISTART is not specified, the time used to calculate the initial delay will be the latest of the start time specified on the TIME control command and the time recorded for any such command. The initial delay is from the start time thus determined to the time of the first record of the initial transmit message.

The method of calculation of the initial delay can be changed from the default previously described by specifying the ISTART operand of the DELAY control statement.

ISTART

Function: The ISTART operand specifies that the start time used to determine an initial delay is only to be taken from the TIME control statement. Any BIND and SDT request units are ignored for purposes of initial delay calculation.

Default: The default is to calculate initial delays as described above for the case of no ISTART operand, taking into account the BIND or SDT RUs, as well as the start time specified on the TIME control statement.

If no specific start time for an initial delay can be determined, no DELAY statement will be generated before the initial TEXT. When ISTART is specified, this will always occur if the TIME control statement specifies START or ALL. If ISTART is not specified, this will happen if both the TIME control statement specifies START or ALL and BIND and SDT RUs are not found in the traced data.

```
NODELAY
```

This command specifies that no DELAY statements are to be inserted into the message generation decks. If neither DELAY nor NODELAY is specified, NODELAY is assumed.

LIMIT and NOLIMIT commands

```
LIMIT nnnnn
```

This command specifies the number of messages to be generated for any one terminal. If the limit is reached, the message generation deck ends.

nnnnn

Function: Specifies that no more than *nnnnn* messages are to be generated.

Format: The *nnnnn* operand is a decimal number from 1 to 65535.

```
NOLIMIT
```

This command specifies that no limit should be imposed on the number of messages to be generated per terminal. If neither LIMIT nor NOLIMIT is specified, NOLIMIT is assumed.

LIST and NOLIST commands

```
LIST
```

This command specifies that all message generation decks should be listed in the SYSPRINT data set as they are created.

```
NOLIST
```

This command specifies that the message generation decks should not be listed in the SYSPRINT data set. If neither LIST nor NOLIST is specified, NOLIST is assumed.

NTWRK and NONTWRK commands

```
NTWRK [{LIST|NOLIST}]
```

This command specifies that the networks submitted as input should be updated to reflect the generated message generation decks.

{LIST|NOLIST}
Function: The LIST operand specifies that the networks should be listed in the SYSPRINT data set as they are updated.

The NOLIST operand specifies that the networks should not be listed in the SYSPRINT data set as they are updated.

Default: NOLIST is the default.

```
NONTWRK
```

This command specifies that the networks submitted as input should not be updated to reflect the message generation decks generated. If neither NTWRK nor NONTWRK is specified, NONTWRK is assumed.

REPORT command

```
REPORT [SUMMARY|FULL]
```

This command specifies whether a summary and a detail report or only a summary report should be printed at the end of processing. “ITPSGEN printed output” on page 249 summarizes the contents of each type of report.

{SUMMARY|FULL}
Function: The SUMMARY operand specifies that only a summary report is to be printed.

The FULL operand specifies that both a summary and detail report are to be printed.

Default: If no REPORT command is specified, only a summary report is produced.

RESP and NORESP commands

```
RESP [{offset|0}][,{maxlen|25}]
```

This command specifies that RESP operands should be added to the CMND statements and the last TEXT statement for each message generated. The RESP data will be up to 25 bytes of the last message received before the next transmit message and following the message for which text is being generated. Received messages shorter than the offset specified will be ignored.

The origin of this data is the beginning of the received data plus the specified offset. The beginning of the received data is the beginning of the request unit (RU) for SNA terminals, and the beginning of the recorded data for all other terminals.

A maximum of 44 characters (including hexadecimal delimiters, hexadecimal digits, text characters, and duplicated delimiter characters) will be included in the deck portion of the RESP data. If no data is received between messages or SNA commands, no RESP data will be added to the TEXT statement or the CMND statement.

Examples of IF statements to use the RESP operands are shown below.

```
0   IF   LOC=RH+0,TEXT='01',ELSE=IGNORE,STATUS=HOLD,SNASCOPE=REQ
1   IF   TEXT=RESP,LOC=RU+0,COND=NE,THEN=B-QUIT,STATUS=HOLD,SNASCOPE=REQ
```

offset
: **Function:** Represents the offset from the beginning of the data to the start of the data to be included in the RESP data.

  Format: The *offset* operand is a decimal number from 0 to 5999.

  Default: The default for *offset* is 0.

maxlen
: **Function:** Represents the maximum number of bytes from the received data that should be included in the RESP data.

  Format: The *maxlen* operand is a decimal number from 1 to 25.

  Default: The default for *maxlen* is 25.

```
NORESP
```

This command specifies that no RESP data should be added to the TEXT or CMND statements for the messages being generated. If neither RESP nor NORESP is specified, NORESP is assumed.

SEQOUT and NOSEQOUT commands

```
SEQOUT [{LIST|NOLIST}]
```

This command specifies that all generated message generation decks and any updated network definitions should be copied to a sequential data set when generation is completed.

{LIST|NOLIST}
: **Function:** The LIST operand specifies that the sequential output records should be listed in the SYSPRINT data set.

  The NOLIST operand specifies that the sequential output records should not be listed in the SYSPRINT data set.

  Default: NOLIST is the default.

```
NOSEQOUT
```

This command specifies that the generated message generation decks and any updated network definitions should not be copied to a sequential data set when generation is completed. If neither SEQOUT nor NOSEQOUT is specified, NOSEQOUT is assumed.

STL and NOSTL commands

```
STL
```

This command is used to determine whether scripts are to be generated in the Structured Translator Language (STL). This command provides you with an option for all types of script generation except CPI-C. CPI-C scripts are exclusively generated in STL. All other types of scripts can be generated in either WSim Scripting Language or STL.

STL specifies that the target language for script generator output is Structured Translator Language (STL).

```
NOSTL
```

NOSTL specifies that the target language for script generator output is WSim Scripting Language. NOSTL is the default value.

TIME commands

```
TIME {ALL}
     {{hhmmss}-{hhmmss}}
      {hhmm}     {hhmm}
      {START}    {END}
```

This command specifies the time limits of the records on the sorted trace data set from which records are to be selected. The first field specifies the start time and the second field specifies the end time. All times are inclusive. If you do not specify a TIME command, all records will be processed.

ALL
: **Function:** The ALL operand specifies that the entire data set should be processed.

  Default: If TIME is not specified, the default is ALL.

START
: **Function:** The START operand indicates that the start time should be the beginning of the input data set.

END
: **Function:** The END operand indicates that the end time should be the end of the input data set.

hhmmss
: **Function:** The *hhmmss* operand is a time entered in hours, minutes, and seconds and in 24-hour format.

hhmm
: **Function:** The *hhmm* operand is a time entered in hours and minutes and in 24-hour format.

Notes:
- Unless TIME ALL is specified, any format of the first operand can be paired with any format of the second operand; for example, TIME 134500-END.
- If you want a startup delay, you must specify a start time on the TIME control command or use the ISTART operand of the DELAY command. See "DELAY and NODELAY commands" on page 242.

* Comment

```
* [data]
```

This command specifies a comment. The command can contain any *data* following the asterisk. The command will be listed with the other input commands before the output reports. The command will be ignored during processing by ITPSGEN.

ITPSGEN message generation decks

The message generation decks generated by ITPSGEN normally have the same names as the terminals for which they are generated. For LUs with multiple sessions, separate decks are generated for each session and a unique name is assigned. The message generation decks consist primarily of TEXT statements. For 3270 terminal types, ERIN, CURSOR, EREOF, JUMP PID=*n*, CLEARPTN, and the various attention identifier keys (ENTER, CLEAR, PF*nn*, PA*n*) are also generated as appropriate. DELAY statements, as well as RESP data on the TEXT statements, can be added if those options were selected.

Messages generated for SNA terminal types from SNA traces are checked to determine if an RH statement needs to be included. Examples of conditions requiring an RH statement to be included are the following:
- Definite response 2 indicated

- End bracket being transmitted
- Chains other than only in chain.

This check is omitted for 3270 terminal types.

In addition, certain SNA commands encountered in the trace input will cause CMND statements to be generated. The commands for which CMND statements will be generated are the following:

- Initiate-Self
- Terminate-Self
- Bid
- Bracket Initiation Stopped
- Quiesce at End of Chain
- Release Quiesce
- Request Shutdown
- Stop Bracket Initiation
- Signal
- Set and Test Sequence Numbers
- Unbind.

Any other commands encountered will be ignored.

A SETSW statement will be generated to turn device switch 7 ON whenever an SNA response is generated. If you want to use these decks as a starting point for your own decks, this statement lets you structure your IF statements to cause the response to be generated properly.

A WAIT statement is inserted at the end of the generated message generation deck. This statement is followed by a BRANCH back to the WAIT. No DELAY is generated after the last message.

The message generation decks are written to the MSGTXT data set. You can use this data set as the MSGDD data set for a subsequent execution of WSim or the Preprocessor.

ITPSGEN updated networks

When the NTWRK statement is specified, ITPSGEN adds PATH statements and PATH operands to the input networks. Added PATH statements have the same name as the message generation deck generated for a terminal or device. PATH operands are added to the LU statements for which the message generation decks are generated.

The entire updated network is written to the NTWRK data set. You can use this data set as the INITDD data set on subsequent execution of WSim or the Preprocessor.

ITPSGEN sequential output format

When you specify the SEQOUT statement, all output that has been written to either the MSGTXT or NTWRK data sets is copied to a single sequential data set defined by the SEQOUT DD statement. If you specified the network update option, each updated network definition is written to the SEQOUT data set, followed immediately by the message generation decks generated for terminals defined by that network. All message generation decks generated for a given network are grouped together in the sequential output.

The message generation decks referred to in the networks but not generated by ITPSGEN are not included in the sequential output data set. If you use the SEQOUT data set as input to the Preprocessor, the MSGDD DD statement for the preprocessor run should be the same as the MSGDD DD statement for ITPSGEN because these message generation decks were placed in this data set during the initial processing of the network by ITPSGEN.

ITPSGEN printed output

All processed commands and all error messages are written to the SYSPRINT data set. Messages indicating the successful initialization of the networks submitted as input are also written to the SYSPRINT data set. If the appropriate LIST options are specified, the generated message generation decks, the updated network definitions, and the sequential output data set can be listed as they are created.

A report is produced at the end of processing that provides information about each input network. This information includes the name of the network, the number of terminals in the network eligible to have message generation decks generated, the number of message generation decks generated, the number of message generation decks for which the limit specified by the LIMIT command was reached, and the number of PATH statements added.

If you specify REPORT FULL, a detailed report is also produced. This report specifies, for each eligible terminal:
- Its name
- Its network name
- The number of messages generated
- The time the first message was generated
- The time the last message was generated
- The PATH name added to the terminal definition in the network definition.

SEQOUT data set

If the SEQOUT control command is specified, ITPSGEN produces a SEQOUT data set as output. If you are generating STL scripts, the SEQOUT data set will contain network definition statements and STL code. The STL must be translated to the WSim Scripting Language before you can run a simulation.

Note: SEQOUT data sets that contain STL source code are structured differently than SEQOUT data sets that contain WSim Scripting Language source code. In SEQOUT data sets that contain STL source, the network definition is surrounded by the @NET and @ENDNET statements to inform the STL Translator to pass it to the Preprocessor. As with non-STL script generation, the network definition will be updated to reflect the generated scripts if the NTWRK control command is specified.

Sample SEQOUT data set

The following is a sample SEQOUT data set.

```
@NET
IDCTSO     NTWRK     UTI=100,LOGDSPLY=BOTH
ITPIDC     PATH      ITPIDC
0          PATH      SGENTXT
* Model network for ITPECHO via IDC.
VAPPL1     VTAMAPPL  APPLID=ITPIDC,INIT=SEC,BUFSIZE=1920
ITPIDC     LU        PATH=(ITPIDC),
                     LUTYPE=LU2
*
@ENDNET
ITPIDC:  MSGTXT
  suspend(3)
  delay(4)
  erin
  cursor(1,27)
  type 'userid'
  cursor(2,7)
  transmit using ENTER
  delay(5)
  erin
  cursor(8,20)
  type 'ABCDEF'
  transmit using ENTER
  delay(21)
  erin
  cursor(5,6)
  transmit using ENTER
  delay(6)
  erin
  cursor(4,14)
  type '9.6;log'
  transmit using ENTER
  delay(3)
  erin
  cursor(4,21)
  type '/d a,1'
  transmit using ENTER
  delay(2)
  erin
  cursor(4,21)
  transmit using PF3
  delay(3)
  erin
  cursor(4,21)
  transmit using PF3
  delay(2)
  erin
  cursor(4,14)
  transmit using PF3
  delay(2)
  erin
  cursor(1,9)
  type 'logoff'
  cursor(2,7)
  transmit using ENTER
  do forever
    wait
  end
  ENDTXT
```

Figure 48. Sample SEQOUT data set

ITPSGEN return codes

At the end of execution, ITPSGEN sets a return code to indicate the status of the execution. ITPSGEN execution ends prematurely for all return codes except 0. ITPSGEN can return the following codes:

Code	Meaning
0	Execution ended successfully.
4	A parameter error occurred.
8	The SYSPRINT data set could not be opened.
12	An invalid control command or control command operand was read from the CTLIN data set.
16	No NTWRK statement was read from the SYSIN data set before the first MSGTXT statement was encountered.
20	At least one of the networks submitted as input would not initialize.
24	Not enough storage was available for execution to continue.
28	The SYSUT3 data set failed to open.
32	The SYSUT2 data set failed to open.
36	The MSGDD data set failed to open.
40	The INITDD data set failed to open.
44	The SYSIN data set failed to open.
48	The INITDD, MSGDD, or RATEDD data set failed to open or not enough storage is available for the IVT control block.
52	The MSGTXT data set failed to open.
56	The TAPEIN data set failed to open.
60	The first record read from the TAPEIN data set was not formatted properly for ITPSGEN.
64	A STOW operation for a member of a partitioned data set failed.
68	An error occurred while writing to an output data set.

Sample output for ITPSGEN

Figures Figure 49 and Figure 50 on page 252 show an example of the report produced by ITPSGEN.

Summary Report

The *trace records eligible* count will appear on the summary report for each network defined in the model network. This field contains a count of the total number of input trace records that were eligible for script generation processing. An example of a summary report containing the *trace records eligible* field is shown below:

```
ITPSGEN GENERATE OUTPUT                    TIME 13.45.09,     JUNE  3, 2002   PAGE     44
                       GENERATION REPORT - SUMMARY

               TERMINALS        TRACE RECORDS        MSGTXTS          LIMIT          PATHS
 NETWORK       ELIGIBLE           ELIGIBLE          GENERATED        REACHED         ADDED

 NET1                 3                 97                 5               0              3
 NET2                 3                  0                 0               0              0
```

Figure 49. Summary Report

Detail Report

The *trace records eligible* count will appear on the detail report for each terminal or resource in the model network. An example of a detail report containing the *trace* *records eligible* field is shown below:

```
 ITPSGEN GENERATE OUTPUT                                    TIME 13.45.09,     JUNE  3, 2002
 PAGE    45
                        GENERATION REPORT - DETAIL

                          TRACE RECORDS     MESSAGES        START           STOP
 TERMINAL      NETWORK      ELIGIBLE        GENERATED       TIME            TIME         PATH

 ITPECHO       NET1               77              37     16:09:05.80     16:09:42.21     ITPECHO
 ITPECHO       NET2                0               0
 NET1TP1       NET1                9               5     08:30:10.94     08:30:20.72     NET1TP1
 NET2TP1       NET2                0               0
 NET1TP2       NET1               11               3     08:30:10.94     08:30:20.73     NET1TP2
 NET2TP2       NET2                0               0
```

Figure 50. Detail Report

Using JCL to run ITPSGEN

The example below shows JCL that you can use to run ITPSGEN.

```
//SGENJOB  JOB
//JOBLIB   DD      DSNAME=WSIM.SITPLOAD,DISP=SHR
//STEP1    EXEC    PGM=ITPSGEN,PARM='CTL'
//RATEDD   DD      DSNAME=WSIM.SITPRTBL,DISP=SHR
//INITDD   DD      UNIT=SYSDA,SPACE=(TRK,(10,10,3))
//MSGDD    DD      DSNAME=WSIM.MSGFILE,DISP=SHR
//SYSUT2   DD      UNIT=SYSDA,SPACE=(TRK,(10,10,3))
//SYSUT3   DD      UNIT=SYSDA,SPACE=(TRK,(10,10,3))
//SYSPRINT DD      SYSOUT=A
//MSGTXT   DD      DSNAME=WSIM.MSGFILE,DISP=SHR
//STLTXT   DD      DSNAME=WSIM.STLIN,DISP=SHR
//NTWRK    DD      DSNAME=WSIM.TESTFILE,DISP=SHR
//SEQOUT   DD      DSNAME=SEQOUT,DISP=SHR
//TAPEIN   DD      UNIT=TAPE,DISP=OLD,VOL=SER=TAPEIN,LABEL=(,NL)
//CTLIN    DD   *
* CONTROL STATEMENTS FOR ITPSGEN SCRIPT GENERATOR *
LIST
DELAY
RESP
SEQOUT
NTWRK
REPORT FULL

/*
//SYSIN    DD   *
SGENNET  NTWRK BUFSIZE=2048,         BUFFER SIZE FOR MESSAGES
               DELAY=A1,             RANDOM AVERAGE DELAY FROM 0->2 TIMES 1
               DISPLAY=(24,80,32,80), PRIMARY & ALTERNATE DISPLAY SIZES
               DLOGMOD=D4A32783,     LOGON MODE TABLE ENTRY FOR LOCAL LU2
               EXTFUN=YES,           3270 EXTENDED FUNCTION SUPPORT PROVIDED
               HEAD='SCRIPT GENERATOR', HEADER FOR ONLINE REPORTS
               INIT=SEC,             WSim WILL INITIATE SECONDARY SESSIONS
               ITIME=1,              PRINT NTWRK INTERVAL REPORT EACH MINUTE
               LUTYPE=LU2,           TYPE OF SIMULATED SNA LOGICAL UNIT
               MAXSESS=(0,1),        ONLY ONE SECONDARY SESSION WITH LU2
               OPTIONS=(DEBUG,CONRATE,MONCMND), SNA SEGM/RU'S & OPCMD
               PATH=(0),             DEFAULT PATH STATEMENT
               RESOURCE=WSIMPLU,     NAME OF APPLICATION UNDER TEST
               RSTATS=YES,           ONLINE RESPONSE TIME STATISTICS
               THKTIME=UNLOCK,       START DELAY CALC AFTER KEYBOARD RESTORE
               USERAREA=4,           SIZE OF STORAGE FOR SCRATCH PAD AREA
               UTI=100               1 SEC. USER TIME INTERVAL FOR DELAYS
*


0        PATH  SGENWAIT
*
```

```
        VTAMAPPL APPLID=WSIM0101,   APPLID DEFINED TO VTAM
              BUFSIZE=2048          BUFFER SIZE OF SIMULATED LU
*
TAF05F01 LU    LOGDSPLY=BOTH,       LOG SCREEN IMAGES
              MSGTRACE=YES          LOG MESSAGE GENERATION TRACE RECORDS
*

SGENWAIT MSGTXT
LOOP     WAIT
         BRANCH LABEL=LOOP
         ENDTXT
/*
```

Using a CLIST to run ITPSGEN

The following example shows a CLIST that you can use to run ITPSGEN.

```
FREE     DDNAME(SYSPRINT RATEDD INITDD MSGDD SYSUT2 SYSUT3)
FREE     DDNAME(MSGTXT STLTXT NTWRK SEQOUT TAPEIN CTLIN SYSIN)
ALLOC    DDNAME(SYSPRINT) SYSOUT(A)
ALLOC    DDNAME(RATEDD)   DATASET('WSIM.SITPRTBL') SHR
ALLOC    DDNAME(INITDD)  UNIT(SYSDA) SPACE(10,10) DIR(3) TRACKS
ALLOC    DDNAME(MSGDD)   UNIT(SYSDA) SPACE(10,10) DIR(3) TRACKS
ALLOC    DDNAME(SYSUT2)  UNIT(SYSDA) SPACE(10,10) DIR(3) TRACKS
ALLOC    DDNAME(SYSUT3)  UNIT(SYSDA) SPACE(10,10) DIR(3) TRACKS
ALLOC    DDNAME(MSGTXT)  DATASET('WSIM.MSGFILE') SHR
ALLOC    DDNAME(STLTXT)  DATASET('WSIM.STLIN')   SHR
ALLOC    DDNAME(NTWRK)   DATASET('WSIM.TESTFILE') SHR
ALLOC    DDNAME(SEQOUT)  DATASET('SEQOUT') SHR
ALLOC    DDNAME(TAPEIN)  DATASET('ITPSGEN.TAPEIN') SHR
ALLOC    DDNAME(CTLIN)   DATASET('ITPSGEN.CTLIN') SHR
ALLOC    DDNAME(SYSIN)   DATASET('ITPSGEN.SYSIN') SHR
CALL     'WSIM.SITPLOAD(ITPSGEN)' 'CTL'
FREE     DDNAME(SYSPRINT RATEDD INITDD MSGDD SYSUT2 SYSUT3)
FREE     DDNAME(MSGTXT STLTXT NTWRK SEQOUT TAPEIN CTLIN SYSIN)
```

Running ITPSGEN from the WSim/ISPF Interface

To run ITPSGEN from the WSim/ISPF Interface, follow these steps:

1. Invoke the WSim/ISPF Interface main panel from ISPF. The method you use to do this depends on how the WSim/ISPF Interface application is installed at your site. If you are not sure how to do this, see your system programmer for assistance.
2. Select option 4 from the WSim/ISPF Interface main panel and press **Enter**. The Generate Message Decks and STL Programs panel is displayed.
3. Select option 3 from this panel and press **Enter**. The Generate Message Decks from Captured Trace Data panel is displayed.
4. Select option 3 from this panel and press **Enter**. The Generate Message Decks from Sorted Trace Data panel is displayed.

   Note: You can also type “GENDECKS” on the WSim/ISPF Interface main panel command line and press **Enter** to display this panel.
5. Fill in the appropriate fields on this panel and press **Enter** to run ITPSGEN.

For more information on the WSim/ISPF Interface, refer to Part 1, “General utilities,” on page 1.

STL translation

Before you can run a script that was generated in STL, you must translate the STL into WSim Scripting Language. If you want to translate all generated scripts during the same STL invocation, point the STL input data set to the data set produced by the script generator SEQOUT DD. Use the sample in “Sample JCL for STL translation” when writing your own JCL to execute the WSim STL Translator.

Sample JCL for STL translation

You can use the following sample when writing your own JCL to execute the WSim STL Translator.

```
//STLJOB   JOB
//*******************************************************************
//* Workload Simulator (WSim)   5655-I39                            *
//*******************************************************************
//*                      STLJOB JCL                                 *
//* Sample JCL to execute the WSim STL Translator (ITPSTL).         *
//*******************************************************************
//STL      EXEC PGM=ITPSTL,REGION=4096K
//STEPLIB  DD  DSN=WSIM.SITPLOAD,DISP=SHR
//PARMDD   DD  DSN=WSIM.PARMDD,DISP=SHR
//RATEDD   DD  DSN=WSIM.SITPRTBL,DISP=SHR
//INITDD   DD  DSN=WSIM.TESTFILE,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//MSGDD    DD  DSN=WSIM.MSGFILE,DISP=SHR
//SEQOUT   DD  DSN=WSIM.STL.SEQOUT,DISP=SHR
//SYSLIB   DD  DSN=WSIM.STLIN,DISP=SHR
//SYSUT1   DD  UNIT=SYSDA,SPACE=(TRK,(10,10,3))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(TRK,(10,10,3))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(TRK,(10,10,3))
//SYSIN    DD  DSN=WSIM.SEQOUT,DISP=SHR
```

Figure 51. Sample JCL for STL translation

Data compression

ITPSGEN now supports the two forms of data compression used in SNA. By default, run-length encoding (RLE) is supported in all traced RU data. When a BIND response is included in the traced RU data, Lempel-Ziv (LZ) encoding is also supported if indicated in the BIND response. To decompress both types of RU data, all RU data must be included in the traced data in order to recreate the original uncompressed RU data.

Decompression errors are reported in the summary and detail reports. The specific type of decompression error is indicated for the transmitted and received records in the detail report.

The following sections include examples of detail and summary reports, and provide possible code values for decompression errors.

Summary report

An example of a summary report containing a *terminals with errors* field is shown below:

```
                                  GENERATION REPORT - SUMMARY

         TERMINALS   TRACE RECORDS    MSGTXTS       LIMIT      PATHS       TERMINALS
NETWORK  ELIGIBLE    ELIGIBLE        GENERATED     REACHED     ADDED       WITH ERRORS
VASGEN        15       10,669             0             0         0          4
```

Problems and possible solutions

Most problems experienced with message generation decks generated by ITPSGEN fall within the following three problem categories. Scripts generated from terminal data traffic captured arbitrarily are prone to these problems.

1. The simulated terminal never sends any data to the host application.
   - *Situation 1.* The real terminal was inactive at the time the data was captured. You can alter the PATH operand to refer to the message generation deck of a terminal sending data to the host application.
   - *Situation 2.* The captured data used to generate the message generation deck did not include the logon or initialization sequence required to start a conversation with the host application. You can code the required sequence in a message generation deck and refer to it by the FRSTTXT operand on the LU statement.
2. The simulated terminal stops sending data to the host application or the data sent is out of synchronization with the host application.
   - *Situation 1.* The timing between messages being generated by WSim is not the same as for the real terminal. You can adjust the terminal's UTI value or you can use the ITPSGEN DELAY command to insert DELAY statements between TEXT statements.
   - *Situation 2.* The messages generated by WSim do not correspond with what the host application is expecting. You can use the ITPSGEN RESP option to add RESP operands to each TEXT statement. The RESP operands and network level IF statements added should synchronize the generated messages with the host application. When the host application sends a common acknowledgment to all transactions, you can add network IF statements to the WSim network definition to wait for this acknowledgment.
   - *Situation 3.* The host application does not unlock the keyboard for a 3270 terminal. If the terminal operator has to unlock the keyboard using the reset key, this same operation must be performed in the WSim network by adding an IF statement to execute a RESET statement. This will unlock the keyboard and allow another message to be generated.
   - *Situation 4.* The host application is expecting the terminal to perform a 3270 selector pen or magnetic stripe reader function. You must add the commands to perform these functions to the message generation decks because their generation is not supported by ITPSGEN.
3. The simulated terminal's initial delay is not what you expect.

   The Script Generator Utility can calculate an initial delay for an LU. The initial delay for each LU is based on the timestamp of the LU, BIND, or SDT [3] (whichever is last to be received before the first data is transmitted) for that LU if the BIND or SDT is included in the trace within the time limits specified by the TIME command. If no BIND or SDT is found for that LU within the time limits, the START time from the TIME command is used for that LU. If no specific START time is specified and no BIND or SDT is found for that LU, no initial delay is generated for that LU.

3. The assumption is that if BIND or SDT is received before anything is transmitted, the terminal must be being acquired by the application, rather than logging on to the application.

The initial delay duration is limited to 65535 times the UTI value specified for the LU.

CPI-C script generation support

The Script Generator Utility (ITPSGEN) enables you to generate CPI-C scripts using data traffic captured from live system runs.

Function overview

The Script Generator Utility (ITPSGEN) can generate CPI-C scripts from traces of CPI-C applications or other applications that produce LU 6.2 line flows.

The procedure for generating CPI-C scripts using ITPSGEN is similar to that used for generating other types of scripts. To generate CPI-C scripts, you must follow these steps:
1. obtain a trace of system activity
2. reformat the captured data using the appropriate utility program
3. sort the reformatted data
4. define a model network for your simulation
5. use ITPSGEN to generate the scripts in STL
6. translate the STL to WSim scripting language using the STL Translator

Before using ITPSGEN to generate CPI-C scripts, you must make minor modifications to the existing script generation JCL or CLIST. Refer to "Changes to JCL and CLISTs" on page 259 for information on the required modifications. You can also create CPI-C scripts by invoking the script generator from the WSim/ISPF Interface.

Note: If you are using the WSim/ISPF Interface, you may need to prep the model network before translating the STL. Otherwise, the wait deck in the model network may not be found during the STL translation.

Tracing considerations

The first step in generating CPI-C scripts is to capture a trace containing LU 6.2 line flows. The trace of system activity can be a VTAM buffer trace, an OS/2 Communications Manager (CM/2) trace, or an IBM Communications Server trace. The trace must be reformatted before it can be used for script generation.

Note: Traces must contain the FMH-5 allocate request for any conversations that are to be simulated. To ensure your trace contains the required allocate requests, activate the trace before establishing any conversations you want to simulate.

A new Work Station Trace Reformatter utility (ITPWSTRF) has been provided with WSim to allow you to reformat OS/2 Communications Manager (CM/2) and IBM Communications Server traces. Refer to Chapter 20, "Work Station Trace Reformatter Utility," on page 275 for more information on ITPWSTRF. The existing VTAM Buffer Trace Reformatter (ITPVTBRF) can be used to reformat VTAM buffer traces. For all tracing options, the reformatted trace file must be sorted before it can be used to generate CPI-C scripts. The files must be sorted in ascending order by resource name, session, date, and time fields.

VTAM buffer trace

You can use a VTAM buffer trace as your source for generating a CPI-C script. The trace may be recorded using either the GTF or the NPM data collection facility. You should set up the trace in the same manner as you would when generating other types of scripts (refer to Chapter 18, "Using the Script Generator Utility," on page 231 for further information). Be sure to request a full buffer trace by specifying the AMOUNT=FULL parameter. After you have traced a scenario that involves CPI-C applications or other applications that produce LU 6.2 line flows, you must reformat the captured trace file using the VTAM Buffer Trace Reformatter (ITPVTBRF). Sort the resulting file using any standard sort program.

OS/2 Communications Manager (CM/2) trace

You can use the OS/2 Communications Manager (CM/2) trace facility to capture an LU 6.2 trace. After capturing a trace, upload the trace file to the host as an EBCDIC TEXT file. Reformat the trace file using the Work Station Trace Reformatter (ITPWSTRF) and then sort the reformatted file. Refer to Chapter 20, "Work Station Trace Reformatter Utility," on page 275 for more information on ITPWSTRF.

IBM Communications Server trace

You can also use the IBM Communications Server trace facility to capture an LU 6.2 trace. After capturing the trace, upload it to the host as an EBCDIC TEXT file. Reformat the trace file using the Work Station Trace Reformatter (ITPWSTRF) and then sort the reformatted file. Refer to Chapter 20, "Work Station Trace Reformatter Utility," on page 275 for more information on ITPWSTRF.

Tracing dependencies and restrictions

Before attempting to generate CPI-C scripts, you should be aware of tracing issues that could affect whether CPI-C scripts accurately represent the intended testing scenario. There is no information in the trace file that differentiates transaction programs or conversations. The CPI-C Script Generator Utility makes the assumption that each session represents a unique transaction program and that each attach request on the session represents the start of a new serial conversation. The Script Generator Utility could produce unexpected results in scripts generated from traces containing multiple transaction programs (TPs) per LU, TPs processing multiple overlapping conversations, or full-duplex sessions. You also should be aware of the circumstances under which the VTAM buffer trace facility may fail to produce complete traces of conversations.

Traces containing multiple TPs or conversations

The CPI-C Script Generation facility considers each unique session captured in a system trace to represent a TP. As a result, scripts might not accurately represent captured traces if the traces include any of the following:

- a TP processing multiple, overlapping conversations
- multiple TPs concurrently active on the same LU
- multiple instances of the same TP

If your trace includes a TP processing multiple conversations, the conversations should be serial, rather than overlapping. If a trace contains a TP processing multiple overlapping conversations or serial conversations using different sessions, the script generated from the trace will not accurately represent the captured trace because the script generator will create a different TP for each conversation.

Traces that include multiple TPs concurrently active on the same LU also can produce unexpected results in generated scripts. Depending on timing, two or more TPs concurrently active on the same LU can have conversations sharing the same session. If this is the case, the generated script will not accurately represent the traced scenario.

If the trace contains multiple instances of the same TP, the resulting script will not accurately represent the original traced scenario. If traces containing multi-instance TPs are used as input to the script generator, you will need to make numerous revisions to the generated scripts to make them accurately represent the original traced scenario.

Note: You will obtain the best results using the CPI-C script generation facility if your trace contains one single-instance TP per LU.

Full-duplex sessions

Trace files that contain one or more full-duplex sessions should be used with caution. WSim simulations that use scripts generated from full-duplex sessions may not accurately reproduce the original traced scenario. However, if a session is identified as full-duplex but used as if it were half-duplex flip-flop, the generated scripts should accurately reproduce the original traced scenario.

VTAM buffer traces

If you will be using VTAM Version 4 Release 4 (or later) buffer traces as your source for generating CPI-C scripts, you should be aware of the circumstances under which using the traces could result in incomplete CPI-C scripts. Buffer traces produced by VTAM Version 4 Release 4 or later do not capture a complete trace of conversations using the APPCCMD interface when the origin and destination are within the same VTAM host. If the application being traced is using the VTAM APPCCMD interface to communicate with a partner APPLID defined on the same VTAM host, only the conversation setup information will be in the VTAM buffer trace data set; all other data sent and received are not captured in the trace. As a result, scripts generated from the trace will be incomplete.

Automatic script generation considerations

As a general rule, scripts produced by the Script Generator Utility should be used as a base. It is unrealistic to expect to play-back hours of live data capture without modification to the generated scripts. If the interdependencies among LUs or TPs is minimal, the modification required should be minor. The best case scenario is LU names need to be updated to match your test system environment. However, as a general rule, there are timing relationships among traced LUs and TPs. For instance, one TP is dependent on receiving an attach request from another TP, or a particular LU makes a database update that is dependent on another LU having made an earlier update to the same field. The script generator creates a script to represent each SNA session as if it is totally independent of all other SNA sessions. Therefore, to accurately reproduce the timing relationships among the generated scripts, logic will need to be manually added to the scripts. This process will require an intimate knowledge of the traced applications.

A good approach to automatic script generation is to limit the trace file to a known transaction rather than tracing all transactions in a system simultaneously. Then add logic to the generated scripts to handle timing relationships for that transaction. Repeat this process for each transaction in the system to be simulated. By limiting the scope of the trace file, timing relationships are much easier to identify.

Network definitions

Before you can use ITPSGEN to generate CPI-C scripts or scripts for other simulation types, you must define a model network. Refer to *Creating WSim Scripts* for detailed information on coding network definition statements.

When creating a network definition for a CPI-C simulation, you must define a single message generation sequence path as path 0. This will be the default path used by any resources for which no script is generated.

A CPI-C APPCLU definition must be specified for each traced resource for which a script will be generated. The APPCLU statement name must match a resource name in the input trace file. There must be at least one TP statement following each APPCLU statement. TP statement names can be selected at your discretion. If automatic network updating is requested, the recommended approach is to specify no operands on the TP statement.

Sample model network

The following is a sample of a network definition intended for use in generating CPI-C scripts. Note that the APPCLU statement names are LU1 and LU2. For a script to be generated for these resources, there must be records in the input trace file with the same resource names.

```
CPICSGEN NTWRK CONRATE=YES,             * Message rates to the console
               OPTIONS=(MONCMND),       * Monitor operator commands
               CPITRACE=VERB,           * Trace CPI-C verbs
               PATH=(0),                * Default path statement
               HEAD='CPI-C NETWORK'
*
*  Model network for CPI-C script generation.
*
0        PATH    SGENTXT
*
LU1      APPCLU
CLIENT   TP
*
LU2      APPCLU
SERVER   TP
*
SGENTXT  MSGTXT
         WAIT
         ENDTXT
```

Figure 52. Network definition for use in generating CPI-C scripts

Changes to JCL and CLISTs

Before using ITPSGEN to generate CPI-C scripts, you must add an STLTXT DD definition to the existing script generation JCL or CLIST. This DD is for an output partitioned data set that will contain the STL source. Each STL MSGTXT will be a member in this partitioned data set. The data set can have the same attributes as those used for the MSGTXT data set. The following sample JCL and CLIST illustrate the addition of the STLTXT DD definition.

Sample JCL

The example below shows JCL that has been modified to enable ITPSGEN to generate CPI-C scripts.

```
//SGENJOB  JOB
//*******************************************************************
//* Workload Simulator (WSim)   5655-I39                           *
//*******************************************************************
//*                     SGENJOB JCL                                 *
//* Sample JCL to execute ITPSGEN.                                  *
//*******************************************************************
//JOBLIB   DD  DSNAME=WSIM.SITPLOAD,DISP=SHR
//STEP1    EXEC PGM=ITPSGEN,PARM='CTL'
//RATEDD   DD  DSNAME=WSIM.SITPRTBL,DISP=SHR
//INITDD   DD  UNIT=SYSDA,SPACE=(TRK,(10,10,3))
//MSGDD    DD  DSNAME=WSIM.MSGFILE,DISP=SHR
//SYSUT2   DD  UNIT=SYSDA,SPACE=(TRK,(10,10,3))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(TRK,(10,10,3))
//SYSPRINT DD  SYSOUT=A
//MSGTXT   DD  DSNAME=WSIM.MSGFILE,DISP=SHR
//STLTXT   DD  DSNAME=WSIM.STLIN,DISP=SHR
//NTWRK    DD  DSNAME=WSIM.TESTFILE,DISP=SHR
//SEQOUT   DD  DSNAME=WSIM.SEQOUT,DISP=SHR
//TAPEIN   DD  UNIT=TAPE,DISP=OLD,VOL=SER=TAPEIN,LABEL=(,NL)
//CTLIN    DD  *
* CONTROL STATEMENTS FOR ITPSGEN SCRIPT GENERATOR *
LIST
SEQOUT
NTWRK
REPORT FULL
/*
//SYSIN    DD  *
CPICSGEN NTWRK CONRATE=YES,             * Message rates to the console
               OPTIONS=(MONCMND),       * Monitor operator commands
               CPITRACE=VERB,           * Trace CPI-C verbs
               PATH=(0),                * Default path statement
               HEAD='CPI-C NETWORK'
*
*  Model network for CPI-C script generation.
*
0        PATH    SGENTXT
*
LU1      APPCLU
CLIENT   TP
*
LU2      APPCLU
SERVER   TP
*
SGENTXT  MSGTXT
         WAIT
         ENDTXT
/*
```

Figure 53. JCL modified to enable ITPSGEN to generate CPI-C scripts

Sample TSO CLIST

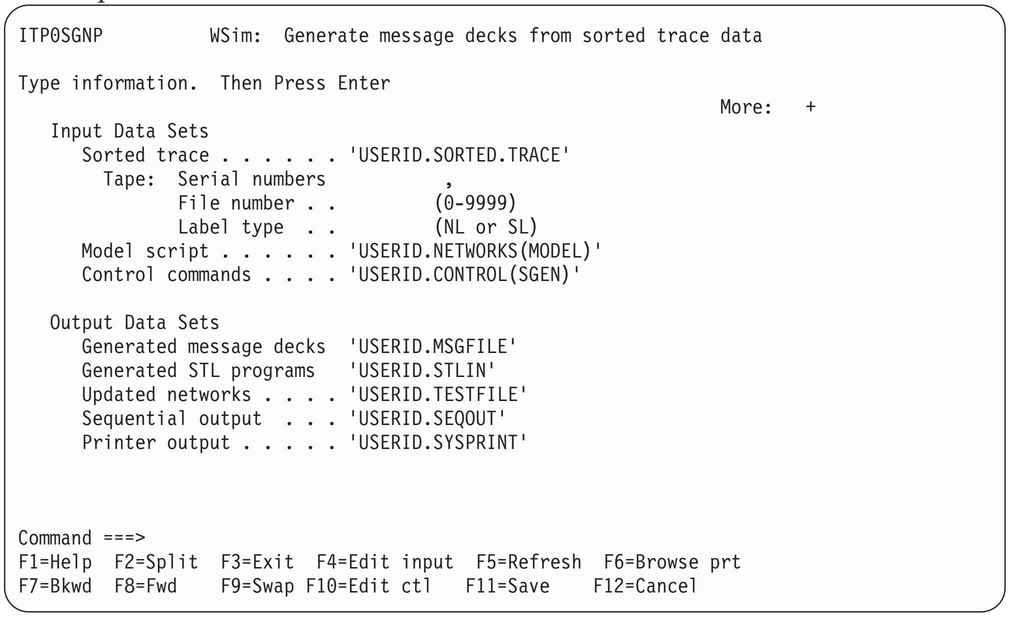
The example below shows a CLIST that has been modified to enable ITPSGEN to generate CPI-C scripts.

```
/*******************************************************************/
/* Workload Simulator (WSim)   5655-I39                            */
/*******************************************************************/
/*                     SGEN CLIST                                  */
/* Sample CLIST to execute ITPSGEN.                                */
/*******************************************************************/
FREE  DDNAME(SYSPRINT RATEDD INITDD MSGDD SYSUT2 SYSUT3)
FREE  DDNAME(MSGTXT STLTXT NTWRK SEQOUT TAPEIN CTLIN SYSIN)
ALLOC DDNAME(SYSPRINT) SYSOUT(A)
ALLOC DDNAME(RATEDD)   DATASET('WSIM.SITPRTBL') SHR
ALLOC DDNAME(INITDD)   UNIT(SYSDA) SPACE(10,10) DIR(3) TRACKS
ALLOC DDNAME(MSGDD)    UNIT(SYSDA) SPACE(10,10) DIR(3) TRACKS
ALLOC DDNAME(SYSUT2)   UNIT(SYSDA) SPACE(10,10) DIR(3) TRACKS
ALLOC DDNAME(SYSUT3)   UNIT(SYSDA) SPACE(10,10) DIR(3) TRACKS
ALLOC DDNAME(MSGTXT)   DATASET('WSIM.MSGFILE')   SHR
ALLOC DDNAME(STLTXT)   DATASET('WSIM.STLIN')     SHR
ALLOC DDNAME(NTWRK)    DATASET('WSIM.TESTFILE')  SHR
ALLOC DDNAME(SEQOUT)   DATASET('WSIM.SEQOUT')    SHR
ALLOC DDNAME(TAPEIN)   DATASET('ITPSGEN.TAPEIN') SHR
ALLOC DDNAME(CTLIN)    DATASET('ITPSGEN.CTLIN')  SHR
ALLOC DDNAME(SYSIN)    DATASET('ITPSGEN.SYSIN')  SHR
CALL  'WSIM.SITPLOAD(ITPSGEN)' 'CTL'
FREE  DDNAME(SYSPRINT RATEDD INITDD MSGDD SYSUT2 SYSUT3)
FREE  DDNAME(MSGTXT STLTXT NTWRK SEQOUT TAPEIN CTLIN SYSIN)
```

Figure 54. A CLIST modified to enable ITPSGEN to generate CPI-C scripts

Changes to the WSim/ISPF Interface

When generating CPI-C scripts using the WSim/ISPF Interface, you must specify a data set to contain the generated STL source. Use the field labeled **Generated STL programs** on panel ITP0SGNP to specify this data set. A member will be created in this data set for each STL MSGTXT that is generated. The data set can have the same attributes as those used for the generated message decks data set. The following is an example of the ITP0SGNP panel as it might look when generating STL scripts.



```
ITP0SGNP          WSim:  Generate message decks from sorted trace data

Type information.  Then Press Enter
                                                              More:   +
   Input Data Sets
      Sorted trace . . . . . . 'USERID.SORTED.TRACE'
        Tape:  Serial numbers           ,
               File number . .         (0-9999)
               Label type  . .         (NL or SL)
      Model script . . . . . . 'USERID.NETWORKS(MODEL)'
      Control commands . . . . 'USERID.CONTROL(SGEN)'

   Output Data Sets
      Generated message decks  'USERID.MSGFILE'
      Generated STL programs   'USERID.STLIN'
      Updated networks . . . . 'USERID.TESTFILE'
      Sequential output  . . . 'USERID.SEQOUT'
      Printer output . . . . . 'USERID.SYSPRINT'



Command ===>
F1=Help  F2=Split  F3=Exit  F4=Edit input  F5=Refresh  F6=Browse prt
F7=Bkwd  F8=Fwd    F9=Swap F10=Edit ctl   F11=Save    F12=Cancel
```

ITPSGEN control commands

This section describes new ITPSGEN control commands specifically for CPI-C scripts.

All of the current control commands are recognized by the CPI-C script generator. However, the RESP and SSCP commands are ignored during generation of CPI-C scripts.

The DELAY command is handled differently when generating CPI-C scripts. Delays are calculated from the time of the last receive record or from the time of the last transmit record, whichever is the most recent. A delay is calculated for all CPI-C verbs that are WSim delimiters (that is, verbs that cause a transmit to VTAM). These verbs are: CMALLC, CMCFM, CMCFMD, CMDEAL, CMFLUS, CMPTR, CMRTS, CMSEND, and CMSERR. The delay is generated in the CPI-C script as an STL SUSPEND statement, and is inserted at the point where the delay was encountered in the trace. When running the generated script, the delays should approximate what actually transpired when the trace was captured. However, no attempt is made to control any delays in responses from the partner. In addition to the SUSPEND statement, a DELAY(0) statement is generated prior to each CPI-C delimiter. This overrides any default delay that may have been set in the network definition, thus preventing a cumulative delay effect.

COMP and NOCOMP commands

```
COMP
```

```
COMP   {ALL  [n] [WARNING | ERROR]}
       {DATA [n] [WARNING | ERROR]}
       {CONV     [WARNING | ERROR]}
```

Default:`COMP ALL 32767 ERROR`

This command specifies the level of comparison to be performed when comparing the actual send data length, and the actual received data, status, and conversation characteristics to the trace file data. Received status comparisons controlled by this command are: send received, confirm received, and conversation deallocated. Conversation characteristic comparisons controlled by this command are: partner LU name, mode name, conversation type, and conversation sync-level. This command also controls whether conversation failures are handled as errors or warnings. Warning conditions result in a message being issued, and the script continues to process. Error conditions result in a message being issued, however, the script is terminated.

COMP ALL[*n*][WARNING | ERROR]
The ALL operand compares send data length, received data, and received status, as well as conversation characteristics. Use *n* to specify the amount of received data to be compared. The range for *n* is 0 to 32767. The default value is 32767. When this control command is specified, the actual send data length is compared to the send data length from the trace. Also, the first *n* bytes of actual received data from each receive verb will be compared against the data from the trace. The actual received data length is also compared to the received data length from the trace. However, if *n* is less than the received data length from the trace, the comparison will only ensure that the actual received data is at least as large as *n*. If *warning*

is specified, the generated script handles conversation failures by issuing a message and continuing with the next statement in the script. If *error* is specified, the generated script handles conversation failures by issuing a message and terminating the script.

COMP DATA[*n*][WARNING | ERROR]
The DATA operand compares only the received data to the trace file data. Use *n* to specify the amount of received data to be compared. The range for *n* is 0 to 32767. The default value is 32767. When this control command is specified, the first *n* bytes of actual received data from each receive verb will be compared against the data from the trace. The actual received data length also will be compared to the received data length from the trace. However, if *n* is less than the received data length from the trace, the comparison will only ensure that the actual received data is at least as large as *n*. If *warning* is specified, the generated script handles conversation failures by issuing a message and continuing with the next statement in the script. If *error* is specified, the generated script handles conversation failures by issuing a message and terminating the script.

COMP CONV[WARNING | ERROR]
The CONV operand compares only the conversation characteristics and received status to the trace file data. If *warning* is specified, the generated script handles conversation failures by issuing a message and continuing with the next statement in the script. If *error* is specified, the generated script handles conversation failures by issuing a message and terminating the script.

```
NOCOMP
```

NOCOMP The NOCOMP control command specifies that no comparisons are to be performed on send data length, received data, status, or conversation characteristics.

FIELD and NOFIELD commands

```
FIELD
```

This command is used to control the creation of composite fields. Composite fields can be optionally generated for the attach request (FMH-5) extension and the send and receive buffers. Creating composite fields refers to mapping the variable length fields in the FMH-5 extension and the send and receive buffers into component pieces and creating STL variables to represent each component. Components that are length fields are scripted in such a way as to support dynamic recalculation when data field components are modified. For example, you can change the fully qualified LU name in the FMH-5 extension by changing the STL variable that represents this value. The FMH-5 length fields affected by the change are automatically recalculated by the script.

FIELD specifies that the variable length fields in the FMH-5 extension and the send and receive buffers should be mapped into composite fields in the generated script. FIELD is the default value.

```
NOFIELD
```

NOFIELD specifies that the variable length fields in the FMH-5 extension and the send and receive buffers should not be mapped into composite fields in the generated script.

NOFIELD Example

The following is an example of the generated script for an FMH-5 extension, send buffer, and receive buffer. This example was generated without composite fields.

```
FMH5_extension = ,
 /* 0000 */  '100702C4E2E4E2C5D90701C4E2E4E2C5D91A11C3E6E2D5C5E3'x||,
 /* EBCDIC:   . . . D S U S E R . . D S U S E R . . C W S N E T     */
 /* 0019 */  'C3C14BC3E6F9F0F0C5F4C9AE08BF98C643000108B619DF7307'x||,
 /* EBCDIC:   C A . C W 9 0 0 E 4 I . . . q F . . . . . . . . .     */
 /* 0032 */  '20E813'x
 /* EBCDIC:   . Y .                                                 */
FMH5_extension_length = length(FMH5_extension)

send_buffer = ,
 /* 0000 */  '001A12210016FF000901E3C5E2E3F1F1C10902C2E4C4F3C2E4'x||,
 /* EBCDIC:   . . . . . . . . . . T E S T 1 1 A . . B U D 3 B U     */
 /* 0019 */  'C4'x
 /* EBCDIC:   D                                                     */
send_length = length(send_buffer)
CMSEND(conversation_ID,send_buffer,send_length,,
       request_to_send_received,return_code)

expected_receive_buffer = ,
 /* 0000 */  '002D12210029FF020300000A0207CD01070F3A29040A0307CC'x||,
 /* EBCDIC:   . . . . . . . . . . . . . . . . . . . . . . . . .     */
 /* 0019 */  '0C171630223D0A0407CD05040000000004050000'x
 /* EBCDIC:   . . . . . . . . . . . . . . . . . . . .               */
```

FIELD Example

The following is an example of the generated script for an FMH-5 extension, send buffer, and receive buffer. This example was generated with composite fields.

```
/* Create FMH-5 extension */
/* Note: If any data field in the FMH-5 extension is modified, all   */
/*       affected length fields will be dynamically recalculated.    */

/* Build access security information */
fm5asipr = '00'x                /* Profile field constant           */
fm5asipw = '01'x                /* Password field constant          */
fm5asiid = '02'x                /* User ID field constant           */
fm5asty1 = fm5asiid             /* User ID security subfield        */
fm5asda1 = 'C4E2E4E2C5D9'x      /* Security subfield data           */
 /* EBCDIC: D S U S E R                                             */
fm5asll1 = ,                    /* Security subfield length         */
           hex(length(fm5asty1)+length(fm5asda1))
fm5asi1  = ,                    /* Composite security subfield      */
           fm5asll1||fm5asty1||fm5asda1
fm5asty2 = fm5asipw             /* Password security subfield       */
fm5asda2 = 'C4E2E4E2C5D9'x      /* Security subfield data           */
 /* EBCDIC: D S U S E R                                             */
fm5asll2 = ,                    /* Security subfield length         */
           hex(length(fm5asty2)+length(fm5asda2))
fm5asi2  = ,                    /* Composite security subfield      */
           fm5asll2||fm5asty2||fm5asda2
```

```
fm5accse = fm5asi1||fm5asi2
fm5lnasi = hex(length(fm5accse))  /* Access Security Info length     */
fm5asi   = fm5lnasi||fm5accse     /* Composite Access Security Info  */

/* Build logical unit of work fields */
fm5fqnam = ,                      /* Fully Qualified Name            */
 /* 0000 */ 'C3E6E2D5C5E3C3C14BC3E6F9F0F0C5F4C9'x
 /* EBCDIC:  C W S N E T C A . C W 9 0 0 E 4 I                       */
fm5lnfqn = hex(length(fm5fqnam))  /* Fully Qualified Name length     */
fm5luwi  = fm5lnfqn||fm5fqnam     /* Composite LUOW ID               */
fm5luwin = 'AE08BF98C643'x        /* LUOW Instance Number            */
 /* EBCDIC: . . . q F .                                              */
fm5luwsn = '0001'x                /* LUOW Sequence Number            */
 /* EBCDIC: . .                                                      */
fm5luow2 = fm5luwin||fm5luwsn     /* Composite LUOW instance/seq     */
fm5lnluw = ,                      /* Logical Unit of Work length     */
           hex(length(fm5lnfqn)+length(fm5fqnam)+length(fm5luow2))
fm5luow1 = fm5lnluw||fm5luwi      /* Composite Logical Unit of Work  */

/* Build conversation correlator */
fm5ccs   = 'B619DF730720E813'x    /* Conversation Correlator data    */
 /* EBCDIC: . . . . . . Y .                                          */
fm5lnccs = hex(length(fm5ccs))    /* Conversation Correlator length  */
fm5cvcor = fm5lnccs||fm5ccs       /* Composite Conv Correlator       */

FMH5_Extension = fm5asi||fm5luow1||fm5luow2||fm5cvcor
 /* 0000    '100702C4E2E4E2C5D90701C4E2E4E2C5D91A11C3E6E2D5C5E3'x||, */
 /* EBCDIC:  . . . D S U S E R . . D S U S E R . . C W S N E T       */
 /* 0019    'C3C14BC3E6F9F0F0C5F4C9AE08BF98C643000108B619DF7307'x||, */
 /* EBCDIC:  C A . C W 9 0 0 E 4 I . . . q F . . . . . . . . .       */
 /* 0032    '20E813'x                                                */
 /* EBCDIC:  . Y .                                                   */
FMH5_extension_length = length(FMH5_extension)

/* Note: If the senddata field is modified, all affected length      */
/*       fields will be dynamically recalculated.                    */
sendid   = '1221'x                /* Probable ID field               */
 /* EBCDIC: . .                                                      */
senddata = ,
 /* 0000 */ '0016FF000901E3C5E2E3F1F1C10902C2E4C4F3C2E4C4'x
 /* EBCDIC:  . . . . . . T E S T 1 1 A . . B U D 3 B U D             */
sendll = hex(length(sendid)+length(senddata)+2,2)
send_buffer = sendll||sendid||senddata /* Send buffer data           */
send_length = length(send_buffer)
CMSEND(conversation_ID,send_buffer,send_length,,
       request_to_send_received,return_code)

/* Note: If the expected_receive_data field is modified, all         */
/*       affected length fields will be dynamically recalculated.    */
expected_receive_id = '1221'x     /* Probable ID field               */
 /* EBCDIC:            . .                                           */
expected_receive_data = ,
 /* 0000 */ '0029FF020300000A0207CD01070F3A29040A0307CC0C17163022'x||,
 /* EBCDIC:  . . . . . . . . . . . . . . . . . . . . . . . . . .     */
 /* 001A */ '3D0A0407CD05040000000000004050000'x
 /* EBCDIC:  . . . . . . . . . . . . . . . .                         */
expected_receive_ll = ,           /* Expected receive buffer length  */
  hex(length(expected_receive_id)+length(expected_receive_data)+2,2)
expected_receive_buffer = ,       /* Expected receive buffer data    */
  expected_receive_ll||expected_receive_id||expected_receive_data
```

HEXON and NOHEXON commands

```
HEXON
```

This command is used to force certain data buffers to be generated in displayable hexadecimal form if they exceed a specified length threshold. The buffers controlled by this command are: *send data receive comparison data error log data* and *FMH-5 extension data*. Optionally, the ASCII and/or EBCDIC translations of the displayable hexadecimal data can be echoed in the form of STL comments. This control command can be useful to facilitate the analysis of the data buffers in the generated script when viewing them in a printout or on a video display screen. Also, if long ASCII data streams are produced in the script, this command can be used to avoid the generation of excessively complex STL statements that exceed the capacity of the STL Translator.

HEXON ALL[ASCII | EBCDIC | BOTH]
The ALL operand specifies that all *send data receive comparison data*, *error log data* and *FMH-5 extension data* will be generated in displayable hexadecimal. You may optionally use ASCII , EBCDIC, or BOTH to specify that the data will be followed with STL comments that echo the data in ASCII, EBCDIC, or both.

HEXON *n* **[ASCII | EBCDIC | BOTH]**
The *n* operand specifies that all *send data*, *receive comparison data*, *error log data* and *FMH-5 extension data* should be generated in displayable hexadecimal if the data length is *n* bytes or more. Data streams less than *n* bytes in length will be generated as a combination of displayable character data and displayable hexadecimal as required for effective display of the data on a video screen. The value for *n* can range from 1 to 32767. You may optionally use ASCII, EBCDIC, or BOTH to specify that for data streams greater than *n* bytes in length, the data will be followed with STL comments that echo the data in ASCII, EBCDIC, or both.

```
NOHEXON
```

NOHEXON NOHEXON is the default for this control command. The NOHEXON command generates all *send data*, *receive comparison data*, *error log data* and *FMH-5 extension data* as a combination of displayable character data and displayable hexadecimal as required for effective display of the data on a video screen.

SENDL and NOSENDL commands

```
SEND
```

```
SENDL [WARNING | ERROR]
```

Default:`SENDL WARNING`

This command is used to determine whether a generated script should report differences in send length compared to the trace file. Reported differences can be treated as warning conditions or error conditions. If *warning* is specified, the generated script handles length differences by issuing a message and continuing with the next statement in the script. If *error* is specified, the generated script handles length differences by issuing a message and terminating the script.

SENDL specifies that the generated script should report differences in the actual send length compared to the trace file.

```
NOSENDL
```

NOSENDL specifies that the generated script should not report differences in the actual send length compared to the trace file.

UCD and NOUCD commands

```
UCD
```

The UCD and NOUCD control commands are used to specify whether user control data should be processed or ignored.

UCD is the default for this control command. The UCD control command specifies that any user control data encountered in the trace should be processed as if they were application data. Also either UCD=YES or UCD=BOTH is added to the TP statement in the model network. If any true application data is encountered, UCD=BOTH is added. Otherwise, UCD=YES is added. Message ITP868I is issued to the SYSPRINT data set the first time user control data is encountered for a given TP. Refer to *WSim Messages and Codes* for more information about the message.

```
NOUCD
```

This control command specifies that any user control data encountered in the trace should be ignored. Also, UCD=NO is added to the TP statement in the model network. Message ITP867I is issued to the SYSPRINT data set the first time user control data is encountered for a given TP. Refer to *WSim Messages and Codes* for more information about the message.

STL translation

CPI-C scripts are generated by ITPSGEN in STL. Before you can run a CPI-C script generated by ITPSGEN, you must translate the STL into WSim scripting language by following the steps given below:

1. The STL input data set must point to the data set produced by the script generation SEQOUT DD. See "Sample JCL for STL translation."
2. Add the CPICVARA and the CPICCON STL include files as members in your STL includes data set (SYSLIB DD) if they are not already in this data set. Both these files are members in the WSim SAMPLE data set. CPICVARA is a new include file added as part of the CPI-C script generation support. See "CPICVARA STL include file" on page 268.

Sample JCL for STL translation

You can use the following sample when writing your own JCL to execute the WSim STL Translator.

```
//STLJOB   JOB
//*********************************************************************
//* Workload Simulator (WSim)   5655-I39                              *
//*********************************************************************
//*                     STLJOB JCL                                    *
//* Sample JCL to execute the WSim STL Translator (ITPSTL).           *
//*********************************************************************
//STL      EXEC PGM=ITPSTL,REGION=4096K
//STEPLIB  DD  DSN=WSIM.SITPLOAD,DISP=SHR
//PARMDD   DD  DSN=WSIM.PARMDD,DISP=SHR
//RATEDD   DD  DSN=WSIM.SITPRTBL,DISP=SHR
//INITDD   DD  DSN=WSIM.TESTFILE,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//MSGDD    DD  DSN=WSIM.MSGFILE,DISP=SHR
//SEQOUT   DD  DSN=WSIM.STL.SEQOUT,DISP=SHR
//SYSLIB   DD  DSN=WSIM.STLIN,DISP=SHR
//SYSUT1   DD  UNIT=SYSDA,SPACE=(TRK,(10,10,3))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(TRK,(10,10,3))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(TRK,(10,10,3))
//SYSIN    DD  DSN=WSIM.SEQOUT,DISP=SHR
```

Figure 55. Sample JCL to execute the WSim STL Translator

CPICVARA STL include file

Before translating the STL, add the following CPICVARA STL include file as a member in your STL includes data set (SYSLIB DD) if it is not already a member of that data set.

```
/********************************************************************/
/* STL variable allocations for CPI-C verb parameters & switches     */
/********************************************************************/
/* String parameters                                                 */
allocate conversation_ID          '1'       /* Conversation ID       */
allocate mode_name                '2'       /* Mode name             */
allocate partner_LU_name          '3'       /* Partner LU name       */
allocate sym_dest_name            '4'       /* Symbolic dest name    */
allocate TP_name                  '5'       /* TP name               */
allocate log_data                 '6'       /* Log data              */
allocate send_buffer              '7'       /* Send buffer           */
allocate receive_buffer           '8'       /* Receive buffer        */
allocate FMH5_extension           '9'       /* FMH-5 extension       */
/********************************************************************/
/* Integer parameters                                                */
allocate conversation_state       'DC1'     /* Conversation state    */
allocate conversation_type        'DC2'     /* Conversation type     */
allocate data_received            'DC3'     /* Data received         */
allocate deallocate_type          'DC4'     /* Deallocate type       */
allocate error_direction          'DC5'     /* Error direction       */
allocate fill                     'DC6'     /* Fill value            */
allocate log_data_length          'DC7'     /* Log data length       */
allocate mode_name_length         'DC8'     /* Mode name length      */
allocate partner_LU_name_length   'DC9'     /* Partner LU name length*/
allocate prepare_to_receive_type  'DC10'    /* Prepare to RCV type   */
allocate receive_type             'DC11'    /* Receive type          */
allocate received_length          'DC12'    /* Received length       */
allocate request_to_send_received 'DC13'    /* Request-to-send rcvd  */
allocate requested_length         'DC14'    /* Requested length      */
allocate return_code              'DC15'    /* Return code           */
allocate return_control           'DC16'    /* Return control        */
allocate send_length              'DC17'    /* Send length           */
allocate send_type                'DC18'    /* Send type             */
allocate status_received          'DC19'    /* Status received       */
allocate sync_level               'DC20'    /* Sync-level            */
allocate TP_name_length           'DC21'    /* TP name length        */
allocate FMH5_extension_length    'DC22'    /* FMH-5 extension length*/
/********************************************************************/
/* Device switches                                                   */
allocate send_received            'SW1'     /* Send token received   */
allocate confirm_received         'SW2'     /* Confirmation req rcvd */
allocate conversation_deallocated 'SW3'     /* Conv deallocated      */
allocate conversation_error       'SW4'     /* Conversation error    */
/********************************************************************/
```

Figure 56. CPICVARA STL include file

VTAM system definitions

Prior to running a WSim simulation using generated CPI-C scripts, make sure each APPC LU to be simulated by WSim is defined to VTAM via APPL statements in the VTAMLST data set. The definition must specify APPC=YES. Each APPC LU must have a unique APPLID name. This is necessary since VTAM associates conversations to an APPLID name and receives attach requests by this name, rather than by the TP name.

Chapter 19. 3270 password masking

Passwords are generally maintained on formatted 3270 screens in unprotected non-display fields. While not visible, the passwords are sent in the clear to host application programs. This enhancement will mask (encrypt or hide using asterisks) potential passwords entered by users of the WSim data capture and script generation utilities and mask their presence in generated WSim scripts, simulation data views, and output reports.

Interactive Data Capture (IDC)

IDC is a VTAM application, which allows 3270 SNA session traffic to be captured for WSim script generation using the log script generation utility ITPLSGEN.

The potential 3270 passwords are masked as follows.

- Transmit and receive data records logged with potential passwords will have all the SNA RU data after the 3270 AID and cursor location bytes masked using asterisks.
- Log Display records logged with potential passwords will be encrypted.

ITPLU2RF

ITPLU2RF reformats a VTAM buffer trace into a WSim log for WSim script generation using the log script generation utility ITPLSGEN.

The potential 3270 passwords are masked as follows.

- Transmit data records logged with potential passwords will have all the SNA RU data after the 3270 AID and cursor location bytes masked using asterisks.
- Log Display records logged with potential passwords will be encrypted.

ITPLSGEN

ITPLSGEN generates a WSim script from a WSim log data set created by IDC, ITPLU2RF, or a WSim simulation run.

The potential 3270 passwords will be masked in the generated script using the following STL code.

```
upnd = 'encrypted_potential_password_data'x
userexit('ITPUMNDX',upnd)
```

The ITPUMNDX message generation exit will ensure the simulated 3270 cursor is in an unprotected non-display field large enough to hold the password and then decrypt the encrypted potential password data into the simulated screen for transmission to the host application program.

WSim simulator ITPENTER

The WSim simulator tokenizes the network definition and scripts into a main storage performance format and executes the scripts to simulate one or more terminals and their simulated operators.

The potential 3270 passwords are masked as follows.

- Transmit data records logged with potential passwords will have just the SNA RU data containing potential 3270 passwords masked using asterisks.
- Log Display records with potential passwords in unprotected non-display fields will be masked using asterisks.
- Display images and transmit data made available for monitoring using the Display Monitor facility will be displayed with potential passwords masked using asterisks.

Loglist utility ITPLL

The WSim loglist utility formats out the WSim log data set for post simulation analysis of the simulation run.

The potential 3270 passwords are masked as follows.

- When Log Display records are processed with encrypted image data from the IDC utility or ITPLU2RF, the potential 3270 passwords will be masked in the output report using asterisks.

Compare utility ITPCOMP

The WSim compare utility compares Log Display records from IDC, ITPLU2RF, or simulation log data sets.

The potential 3270 passwords are masked as follows.

- When Log Display records are processed with encrypted image data from the IDC utility or ITPLU2RF, the potential 3270 passwords will be masked in the image compare buffer using asterisks. Since the WSim simulator generates Log Display records with the potential 3270 passwords also masked using asterisks, all potential 3270 passwords will contain asterisks for the length captured or entered by the simulation script.

ITPGNKYZ utility to generate encryption key/IV USERMOD

The ITPGNKYZ utility generates an SMP/E USERMOD to set a site unique encryption key and initialization vector (IV) value. The following JCL is used to execute the ITPGNKYZ utility. The IN and OUT DDs are for RECFM=FB, LRECL=80 data sets.

```
//GNKEYZAP JOB
//STEP1    EXEC PGM=ITPGNKYZ
//STEPLIB  DD   DSN=WSIM.SITPLOAD,DISP=SHR
//IN       DD   *
kykykykykykykykyivivivivivivivivxiv
/*
//OUT      DD   DSN=KEYIV.USERMOD,DISP=SHR
```

Where kykykykykykykyky is an eight-byte key value in hex and iviviviviviviviv is an eight-byte IV value in hex, i.e. a total of thirty-two hex characters.

The following return codes are set by ITPGNKYZ.

0	OK
8	Invalid Key or IV Value
12	OPEN Error on Input File
16	READ Error on Input File
20	OPEN Error on Output File
24	WRITE Error on Output File

Chapter 20. Work Station Trace Reformatter Utility

Work Station Trace Reformatter Utility (ITPWSTRF) is a REXX exec that reformats OS/2 Communications Manager (CM/2) and IBM Communications Server LU 6.2 traces into TIR formatted records for processing by the WSim Script Generator (ITPSGEN). The steps for using OS/2 Communications Manager (CM/2) and IBM Communications Server traces as script generation source files are as follows:

1. Capture an LU 6.2 trace using either the OS/2 Communications Manager (CM/2) trace facility or the IBM Communications Server trace facility.
2. Upload the trace output file to your host system as an EBCDIC TEXT file.
3. Run the Work Station Trace Reformatter utility (ITPWSTRF) against the uploaded trace file.
4. Sort the ITPWSTRF output file in ascending order by resource name, session number, date, and time fields.
5. Run the Script Generator utility (ITPSGEN) against the sorted file.

Trace output format requirements

ITPWSTRF can only be used to reformat trace output produced by the OS/2 Communications Manager (CM/2) or IBM Communications Server trace facilities. Examples of the format required for the trace output files are provided below.

Note: ITPWSTRF will only process trace records in the formats listed below. The trace record formats are subject to change as the OS/2 Communications Manager (CM/2) and IBM Communications Server products evolve over time.

If the Work Station trace does not match one of the following formats, such that ITPWSTRF cannot be used, you should use a VTAM buffer trace and ITPVTBRF to provide input for CPI-C script generation.

Example: CM/2 trace record

```
"TRACE COPIED 06/30/2002 13:09:05.06""
<==SEND=====  IBMTRNET  #00 40003745100204  0C400774E27E37BF          13:08:08.16
      #:009F TH:2F0001020001 RH:6B8100
    31001307 B0B05033 01808686 80010602      <1.....P3..ff....>
    00000000 0000001C 23000010 E4E2C9C2      <........#...USIB>

===RECV====>  IBMTRNET  #00 40003745100204  0C400774E27E37BF          13:08:10.44
      #:0068 TH:2F0002010001 RH
    00000000 00000010 23000000 25000902       <........#...%...>
```

Figure 57. Example: CM/2 trace record

Executing ITPWSTRF under MVS

To execute ITPWSTRF under MVS, specify the following:

ITPWSTRF *input_file output_file [INVERT]*

Where:

- *input_file* is the input file name
- *output_file* is the output file name
- `INVERT` causes the reformatter to switch the origin and destination LU names

Notes:

- The *input_file* and *output_file* parameters are required.
- The input and output files must be different physical files.
- If the file name is not specified in quotes, the TSO userid is added as the high-level qualifier.

Chapter 21. Generating STL from TCP/IP traces

You can generate STL programs from a TCP/IP trace with the TCP/IP Trace STL Generation Utility (ITPIPGEN). ITPIPGEN reads a data set that contains TCP/IP trace records and processes those records. The records contain HTTP messages that are exchanged between a server and client. An STL program is created, which replicates the communication that occurred between the server and client.

ITPIPGEN can also generate a network definition that can be used to run message deck. You can create the message deck from the generated STL program.

The generated STL program imitates the processing that is performed by the client. The program sends the client messages that are obtained from the trace to the server port and waits to receive a message from the server.

The program can include logic to compare the HTTP header status line in the message that is returned from the server with the corresponding message from the trace. You can invoke the logic by specifying the verification option. If the messages match, the transaction within the test case is considered successful. The following processes continue until all the client and server messages from the trace are processed:
- Sending a client message to the server.
- Receiving the server response message.
- Comparing the server response message with the corresponding server message from the trace, if the verification option is specified.

Setting up ITPIPGEN

To run ITPIPGEN under MVS, you need to allocate or use an existing allocation for the following data sets:

STL programs
This data set contains the generated STL programs. This data set must be partitioned and allocated as fixed block, variable block, or variable with a record length of at least 71 bytes and a block size compatible with the record length. The space that is needed for this data set depends on the number of user actions and amount of data that is generated in the program. Initially, allocate at least five cylinders of 3390 DASD or equivalent space for this data set. This data set must be cataloged.

Network definitions
If required, ITPIPGEN can create a network definition to run the message deck from the generated STL program. This data set must be partitioned and its block size must be a multiple of 80. This data set is optional.

ITPIPGEN commands
This data set contains the ITPIPGEN commands that are used to control the script generation process. You can allocate this data set as a fixed block data set with a record length of 80 bytes and a block size compatible with the record length. The space that is needed for this data set depends on the number of ITPIPGEN commands that are entered.

The input to ITPIPGEN is a sequential data set that contains TCP/IP trace records. You can create this data set by using the WSim TCP/IP Trace Utility program ITPIPTRX.

Running ITPIPGEN

When you set up the data sets, you can then start the ITPIPGEN utility. To do this, you need to define the ITPIPGEN job stream and specify the ITPIPGEN execution parameters. You can also run ITPIPGEN by using the WSim/ISPF Interface.

Using JCL to run ITPIPGEN

The following example shows the JCL to run ITPIPGEN as an MVS batch job.

```
//ITPIPGEN JOB
//ITPIPGEN EXEC PGM=ITPIPGEN
//STEPLIB  DD DSN=WSIM.SITPLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSIN    DD DSN=WSIM.ITPIPGEN.COMMANDS,DISP=SHR
```

Using a CLIST to run ITPIPGEN

The following example shows the CLIST commands to run ITPIPGEN under TSO.

```
ALLOC DDNAME(SYSPRINT) SYSOUT(A)
ALLOC DDNAME(WSTSKLIB) DSNAME('WSIM.SITPLOAD') SHR
ALLOC DDNAME(SYSIN)    DSNAME('WSIM.ITPIPGEN.COMMANDS') SHR
CALL  'WSIM.SITPLOAD(ITPIPGEN)'
FREE DDNAME(SYSPRINT WSTSKLIB SYSIN)
```

Running ITPLSGEN from the WSim/ISPF Interface

To invoke ITPIPGEN from the WSim/ISPF Interface, follow these steps:

1. Invoke the WSim/ISPF Interface main panel from ISPF. The method that you use to do this depends on how the WSim/ISPF Interface application is installed at your site. If you are not sure how to do this, see your system programmer for assistance.
2. Select option 5 from the WSim/ISPF Interface main panel and press **Enter**. The Generate Message Decks and STL Programs panel is displayed.
3. Select option 4 from this panel and press **Enter**. The Generate an STL Program from a TCP/IP Trace panel is displayed.

   Note: You can also type “TCPSGEN” on the WSim/ISPF Interface main panel command line and press **Enter** to display this panel.
4. Fill in the appropriate fields on this panel and press Enter to run ITPIPGEN.

For more information on the WSim/ISPF Interface, see Chapter 2, “Running WSim with the WSim/ISPF Interface,” on page 5

Specifying execution parameters

You can specify the following execution parameters for ITPIPGEN:

PRTLNCNT=*nnn*
: Specifies the maximum number of lines to be printed on a page of output before a new page starts. The value for *nnn* is an integer from 35 to 255. The default value for *nnn* is 60.

ROUTCDE=(*n*,*n*,...)
Specifies the system message routing codes to be used when ITPIPGEN writes messages to the operator. Each *n* is a system routing code that defines a console destination for every WTO and WTOR message that ITPLSGEN writes. *n* can be an integer from 1 to 16. The default value for the ROUTCDE parameter is 8.

Using control commands

You can enter control commands to complete the following tasks:

- Specify the IP address of the client and the port number for the server
- Define the program name and STL member name
- Specify the input and output data set names
- Generate a network definition
- Start the utility
- End the utility

Note: You must enter the following input commands:

- IP
- PORT
- MSGTXT
- INPUT
- STLOUT
- RUN
- END

All other commands are optional.

Entering control commands

You can enter a control command in any position in the input record of the SYSIN DD data set. These commands cannot extend past column 71 and cannot be continued. Operands can be separated with only commas. Although you can separate a command and its operand by more than one blank space, you must enter at least one blank space between the command and the operand. You cannot enter blank spaces between operands; operands that follow the space are interpreted as comments. To enter a comment on a line with operands, insert at least one space in between the operands and the comment.

You might also have comment lines. You can do this by entering an asterisk (*) in the first column.

If you enter a command more than once within the same run, ITPIPGEN uses the last valid one entered. You can enter multiple RUN commands in the SYSIN DD data set or from the WSim operator console. Successful generation of STL following a RUN command clears all the commands entered; a RUN command that results in an error does not clear the commands.

All commands can be abbreviated to any shorter length. All of these commands, except for NTWRK, can be shortened to one letter. The abbreviated command for NTWRK is NT. The following examples are identical.

```
STLOUT 'WSIM.TEST.STL'
S 'WSIM.TEST.STL'
STL 'WSIM.TEST.STL'
```

Specifying the client IP address

You can specify the IP address of the client by using the IP command. This command is shown below.

IP *ipaddr*

ITPIPGEN uses the IP address that is specified with the IP command to search the input TCP/IP trace data set for messages that are exchanged between the client IP address and the server port. *ipaddr* must be a valid IPV4 or IPV6 address.

Specifying the server port

You specify the port number that is used by the server by using the PORT command. This command is shown below.

PORT *portnum*

ITPIPGEN uses the port number that is specified with the PORT command to search the input TCP/IP trace data set for messages that are exchanged between the client IP address and the server port.

Defining the STL program name

You can define the name of the generated STL program with the MSGTXT command. This command is shown below.

MSGTXT *msgtxtid*

msgtxtid must be 1 to 8 alphanumeric or special ($,@,_,?,#) characters, where the first character is non-numeric.

The name cannot be an STL reserved word or begin with $INC, $LA, or $SET, which are reserved labels in STL. Also, you cannot use the same name as the STL program trace name.

Specifying input and output

You can specify the input and output data sets by using the INPUT, STLOUT, and NETOUT commands.

To specify the input TCP/IP trace data set name, code the INPUT command. This command is shown below.

INPUT *dsname*

dsname is the data set name of the TCP/IP trace data set. This name can be from 1 to 36 characters long and can be enclosed in single or double quotation marks. If the data set name that you specify has embedded spaces in it, you must enclose the name in quotation marks. When you use partitioned data sets under MVS, specify the member name within parentheses after *dsname*.

To specify the output STL data set name, code the STLOUT command. This command is shown below.

STLOUT *dsname*

dsname is the data set name that contains the generated STL. This name can be from 1 to 36 characters long and can be enclosed in single or double quotation marks. If the data set name you specify has embedded spaces in it, you must enclose the name in quotations. When you are using partitioned data sets under MVS, the member name can optionally be specified within parentheses after *dsname*. The default member name for output partitioned data set is the *msgtxtid* that is specified on the MSGTXT command.

If you require a network definition to be generated, specify the NETOUT command as shown below.

NETOUT *dsname*

dsname is the data set name that contains the generated network definition. This name can be from 1 to 36 characters long and can be enclosed in single or double quotation marks. If the data set name you specify has imbedded spaces in it, you must enclose the name in quotation marks. When you use partitioned data sets under MVS, the member name can optionally be specified within parentheses after *dsname*. The default member name for output partitioned data set is the *ntwrkid* that is specified on the NTWRK command or *msgtxtid* that is specified on the MSGTXT command if the NTWRK command is not specified.

Defining the network name

If you request the generation of a network definition by using the NETOUT command, you can use the NTWRK command to define the name of the network. This command is shown below:

NTWRK *ntname*

ntname must be 1 to 8 alphanumeric or special ($,@,_,?,#) characters, where the first character is non-numeric.

Verifying server responses

You can request that the generated STL includes logic to verify that the status line in a response from the server at test execution matches the status line in the corresponding message in the TCP/IP trace that is used to generate the STL. To do this, use the VERIFY command that is shown below:

VERIFY

When you enter the VERIFY command, ITPIPGEN generates STL that contains logic to check that the status line in a server response matches the corresponding server message in the TCP/IP trace.

To generate STL without server response verification, use the NOVERIFY command as shown below.

NOVERIFY

If you do not specify either the VERIFY or NOVERIFY command, ITPIPGEN generates STL that includes logic to verify responses from the server.

Starting ITPIPGEN

When you finish entering your control commands, you then start the ITPIPGEN utility with the RUN command. This command is shown below.

```
RUN
```

The RUN command specifies that all commands are entered and that script generation is to begin. After script generation, ITPIPGEN resets everything to their default values.

You can only enter RUN after you specify IP, PORT, MSGTXT, INPUT, and OUTSTL for that run. If you enter RUN before you specify these commands, ITPLSGEN does not process any of your commands. This lets you enter the required commands without losing what you previously entered.

Ending ITPIPGEN

The last control command that you enter is the END command. This command is shown below.

```
END
```

The END command tells ITPIPGEN to come to a normal completion. No further processing occurs. Any commands that are entered after the last RUN command are ignored.

Part 3. Appendixes

Notices

This information was developed for products and services that are offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Trademarks and service marks

Glossary

This glossary includes terms and definitions from the *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems*, GC20-1699-6. Further definitions are from the following volumes and reports. The symbols follow the definitions to which they refer.

- The *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.
- Definitions from draft proposals and working papers under development by the International Standards Organization, Technical Committee 97, Subcommittee 1 are identified by the symbol (TC97).
- Definitions from draft international standards, draft proposals, and working papers in development by the ISO/TC97/SC1 are identified by the symbol (T), indicating final agreement has not yet been reached among participating members.
- Definitions from the *CCITT Sixth Plenary Assembly Orange Book, Terms and Definitions* and working documents published by the International Consultative Committee on Telegraph and Telephone of the International Telecommunication Union, Geneva, 1980 are identified by the symbol (CCITT/ITU).
- Definitions from published sections of the *ISO Vocabulary of Data Processing*, developed by the International Standards Organization, Technical Committee 97, Subcommittee 1 and from published sections of the *ISO Vocabulary of Office Machines*, developed by subcommittees of ISO Technical Committee 95, are indicated by the symbol (ISO).

A

AID Attention identifier.

API Application program interface.

application program interface (API)
(1) The formally defined programming language interface between an IBM system control program or licensed program and its user. (2) The interface through which an application program interacts with an access method. In VTAM, it is the language structure used in control blocks so that application programs can reference them and be identified to VTAM.

attention identifier (AID)
A code that the terminal sends in the inbound data stream to identify the operator action or structured field function that caused the data stream to be sent to the application program. An AID is always sent as the first byte of the inbound data stream. Structured fields in the data stream may also contain an AID.

available
In VTAM, pertaining to a logical unit that is active, connected, enabled, and not at its session limit.

B

basic transmission unit (BTU)
In SNA, the unit of data and control information passed between path control components. A BTU can consist of one or more path information units (PIUs).

bind In SNA, a request to activate a session between two logical units (LUs).

BTU Basic transmission unit.

C

chain A group of logically linked records, for example, an SNA message.

character set
(1) A defined collection of characters in a loadable or nonloadable set selected by means of a local character set identifier. (2) An attribute type in the extended field and character attributes. (3) An attribute passed between session partners in the Start Field Extended, Modify Field, and Set Attribute orders.

Common Programming Interface for Communications (CPI-C)
In WSim, an application programming

interface (API) used to perform program-to-program communications using LU type 6.2 communication protocols. An evolving application programming interface (API), embracing functions to meet the growing demands from different application environments and to achieve openness as an industry standard for communications programming. CPI-C provides access to interprogram services such as (a) sending and receiving data, (b) synchronizing processing between programs, and (c) notifying a partner of errors in the communication.

CPI-C Common programming interface for communications.

D

data flow control (DFC)
In SNA, a request/response unit (RU) category used for requests and responses exchanged between the data flow control layer in one half-session and the data flow control layer in the session partner.

data set
The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

data set members
Members of partitioned data sets that are individually named elements of a larger file that can be retrieved by name.

DBCS Double-byte character set.

ddname
Data definition name.

duplex
In data communication, pertaining to a simultaneous two-way independent transmission in both directions. Synonymous with full duplex. (A) Contrast with half duplex.

E

EBCDIC
Extended binary-coded decimal interchange code.

end bracket (EB)
In SNA, the value (binary 1) of the end bracket indicator in the request header (RH) of the first request of the last chain of a bracket; the value denotes the end of the bracket.

end-of-transmission character (EOT)
The specific character, or sequence of characters, that indicates no more data.

EOT End-of-transmission character.

event (1) An occurrence of significance to a task; typically, the completion of an asynchronous operation, such as an input/output operation. (2) In WSim, a named indicator/flag which can be used for communications among terminal scripts.

extended binary-coded decimal interchange code (EBCDIC)
A coded character set of 256 8-bit characters.

extended field attribute
Additional field definition to the field attribute that controls defining additional properties such as color, highlighting, character set, and field validation. The extended field attribute is altered by information passed in the Start Field Extended and Modify Field orders.

F

facility
(1) An operational capability, or the means for providing such a capability. (T) (2) A service provided by an operating system for a particular purpose; for example, the checkpoint/restart facility.

FID SNA format identification.

File Transfer Protocol (FTP)
In the Internet suite of protocols, an application layer protocol that uses TCP and Telnet services to transfer bulk-data files between machines or hosts.

FM Function management.

format identification (FID) field
In SNA, a field in each transmission header (TH) that indicates the format of the TH; that is, the presence or absence of certain fields. TH formats differ in accordance with the types of nodes between which they pass.

FTP File transfer protocol.

H

half duplex
In data communication, pertaining to an alternate, one way at a time, independent transmission(A); Contrast with duplex.

I

I/O Input/output.

IMS/VS
Information Management System/Virtual Storage.

Information Management System/Virtual Storage (IMS/VS)
A general purpose system that enhances the capabilities of OS/VS for batch processing and telecommunication and allows users to access a computer-maintained data base through remote terminals.

input/output (I/O)
(1) Pertaining to a device whose parts can perform an input process and an output process at the same time. (2) Pertaining to a functional unit or channel involved in an input process, output process, or both, concurrently or not, and to the data involved in such a process. ***Note:*** *The phrase input/output may be used in place of input/output data, input/output signals, and input/output process when such a usage is clear in context.* (3) Pertaining to input, output, or both.

Interactive System Productivity Facility (ISPF)
An IBM licensed program that serves as a full-screen editor and dialogue manager. Used for writing application programs, it provides a means of generating standard screen panels and interactive dialogues between the application programmer and terminal user.

intermessage delay
The elapsed time between receipt of a system response at a terminal and the time when a new transaction is entered. Synonymous with think time.

IOB Input/output control block.

ISPF Interactive System Productivity Facility.

J

JCL Job control language.

job control language (JCL)
A problem-oriented language designed to express statements in a job that are used to identify the job or describe its requirements to an operating system. (A)

L

Log Compare Utility
A utility that enables WSim to compare 3270 display records from two log data sets and report the differences.

logic test
In WSim, a conditional test on an input or output message, a counter, or other item using the WSim IF statement. The IF actions can be used to control the message generation process.

logical unit (LU)
(1) A port through which a user gains access to the services of a network. (2) In SNA, a port through which an end user accesses the SNA network and the functions provided by system services control points (SSCPs). An LU can support at least two sessions—one with an SSCP and one with another LU—and may be capable of supporting many sessions with other logical units.

Loglist Utility
A utility that enables WSim to produce a formatted report of the log data set.

LU Logical unit.

M

MDT Modified data tag.

message generation
In WSim, the process of executing WSim statements that generate messages from the resources being simulated by WSim.

message generation statements
The collection of statements that define the actions to be performed by WSim, including message generation and logic testing.

MF Modify field.

modified data tag (MDT)
(1) An indicator, associated with each input or output/input field in a displayed record, that is set ON when data are keyed into the field. The modified data tag is maintained by the display device

and can be used by the program using the file. (2) In 3270, a bit in each input field that, when set, causes that field to be transferred to the host system.

modify field (MF)
A 3270 data stream order that specifies the field and extended field attributes to be modified without having to respecify all attributes of the field.

module
A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to, or output from, an assembler, compiler, linkage editor, or executive routine. (A)

MTRC
Message generation trace record.

Multiple Virtual Storage (MVS)
An IBM licensed program whose full name is the Operating System/Virtual Storage (OS/VS) with Multiple Virtual Storage/System Product for System/370*. It is a software operating system controlling the execution of programs.

MVS Multiple Virtual Storage.

N

NCB Network control block.

NetView® Performance Monitor (NPM)
An IBM licensed program that collects, monitors, analyzes, and displays data relevant to the performance of a VTAM telecommunication network. It runs as an online VTAM application program.

network control (NC)
In SNA, an RU category used for requests and responses exchanged for such purposes as activating and deactivating explicit and virtual routes and sending load modules to adjacent peripheral nodes.

network control block (NCB)
A WSim control block containing information about simulated networks.

network definition statements
A collection of statements that define the network configuration WSim uses when processing the message generation source statements.

node (1) In SNA, an endpoint of a link or junction common to two or more links in a network. Nodes can be distributed to host processors, communication controllers, cluster controllers, or terminals. Nodes can vary in routing and other functional capabilities. (2) In VTAM, a point in a network defined by a symbolic name.

O

OEF Origin element field.

operating system (OS)
Software that controls the execution of programs. An operating system may provide services such as resource allocation, scheduling, input/output control, and data management. ***Note:*** *Although operating systems are predominantly software, partial or complete hardware implementations are possible.* (A)

origin element field (OEF)
In SNA, a field in an FID4 transmission header that contains an element address, which combined with the subarea address in the origin subarea field (OSAF), gives the complete network address of the originating network addressable unit (NAU).

OS Operating system.

P

PA Program attention.

partitioned data set (PDS)
A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

path information unit (PIU)
In SNA, a message unit consisting of a transmission header (TH) alone, or of a TH followed by a basic information unit (BIU) or a BIU segment.

PF Program function.

PIU Path information unit.

programmed symbols (PS)
In the 3270 Information Display System, an optional feature that stores up to six user-definable, program-loadable character sets of 190 characters each in

terminal read/write storage for display or printing by the terminal.

PS Programmed symbols.

R

record (1) A set of data treated as a unit (TC97); for example, in stock control, each invoice could constitute one record. (2) In VTAM, the unit of data transmission for record-mode. A record represents whatever amount of data the transmitting node chooses to send. (3) In Series/1*, a portion of a data set accessed at the logical level (GET/PUT).

request/response header (RH)
In SNA, control information preceding a request/response unit (RU), that specifies the type of RU (request unit or response unit) and contains control information associated with that RU.

request/response unit (RU)
In SNA, a generic term for a request unit or a response unit.

request unit (RU)
(1) In SNA, a message unit that contains control information, such as a request code, or function management (FM) headers, end-user data, or both. (2) In DPCX, the smallest unit of data or control information.

resource
(1) Any facility of the computing system or operating system required by a job or task, and including main storage, input/output devices, the processing unit, data sets, and control or processing programs. (2) In the NetView program, any hardware or software that provides function to the network.

Response Time Utility
A utility that enables WSim to analyze response times for activities on the log data set.

response unit (RU)
In SNA, a message unit that acknowledges a request unit; it may contain prefix information received in a request unit. If positive, the response unit may contain additional information (such as session parameters in response to BIND session), or if negative, contains sense data defining the exception condition.

return code
A code used to influence the execution of succeeding instructions. (A)

RH Request header or response header.

RU Request unit or response unit.

S

script See WSim script.

session control (SC)
In SNA, (1) One of the components of transmission control. Session control is used to purge data flowing in a session after an unrecoverable error occurs, to resynchronize the data flow after such an error, and to perform cryptographic verification. (2) A request unit (RU) category used for requests and responses exchanged between the session control components of a session and for session activation and deactivation requests and responses.

SI Shift In. Used with DBCS. This is the X'0F' character that ends DBCS data.

SNA Systems Network Architecture.

SO Shift Out. Used with DBCS. This is the X'0E' character that begins DBCS data.

SS Start-stop.

STL Structured Translator Language.

STL Translator
In WSim, a utility that acts as the STL translator and translates STL statements into message generation source statements.

Structured Translator Language (STL)
A set of conventions and rules for writing syntactically allowable statements that will create message generation source statements.

Systems Network Architecture (SNA)
The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through and controlling the configuration and operation of networks.

T

TH Transmission header.

think time
The elapsed time between receipt of a system response at a terminal and the time when a new transaction is entered. Synonym for intermessage delay.

time sharing option (TSO)
An optional configuration of the operating system that provides conversational time sharing from remote stations in a network using VTAM.

TP Transaction program.

TPF Transmission priority field.

transaction program (TP)
In WSim, a transaction program is any program that uses LU type 6.2 communication protocols to communicate with another program. Transaction programs are implemented in WSim using the CPI-C application program interface.

transmission group (TG)
In SNA, a group of links between adjacent subarea nodes, appearing as a single logical link for routing of messages. A transmission group may consist of one or more SDLC links (parallel links) or of a single System/370 channel.

transmission header (TH)
In SNA, control information, optionally followed by a basic information unit (BIU) or a BIU segment, that is created and used by path control to route message units and to control their flow within the network.

TSO Time sharing option.

TVT WSim vector table.

V

Virtual Telecommunications Access Method (VTAM)
An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

VTAM
Virtual Telecommunications Access Method.

W

WCC Write control character.

window
(1) In SNA, synonym for pacing group. (2) On a visual display terminal, a small amount of information in a framed-in area on a panel that overlays part of the panel. (3) In data communication, the number of data packets a data terminal equipment (DTE) or data circuit-terminating equipment (DCE) can send across a logical channel before waiting for authorization to send another data packet.

window size
The specified number of frames of information that can be sent before receiving an acknowledgment response.

Workload Simulator (WSim)
IBM program product to simulate terminals and networks. It enables the user to test system performance and evaluate network design.

write control character (WCC)
(1) A control character that follows a write command in the 3270 data stream and provides control information for executing display and printer functions. (2) A character used in conjunction with a write-type command to specify that a particular operation, or combination of operations, is to be performed at a display station or printer.

write-to-operator (WTO)
An optional user-coded service that enables the writing of a message to the system console operator that informs the operator of errors and unusual system conditions that may need correcting.

write-to-operator-with-reply (WTOR)
An optional user-coded service whereby a message may be written to the system console operator informing him of errors and unusual conditions that may need correcting. The operator must key in a response.

WSim Workload Simulator.

WSim network
The set of statements defining an entire WSim network, including both the network definition statements and the

message generation source statements. Should not be confused with a packet switching network.

WSim script
The set of statements defining an entire WSim network, including both the network definition statements and the message generation source statements.

WTO Write-to-operator.

WTOR
Write-to-operator-with-reply.

X

XMIT Transmit.

Bibliography

The following manuals provide additional information about the definition and operation of networks simulated by WSim:

WSim library

WSim User's Guide, SC31-8948

WSim Messages and Codes, SC31-8951

Creating WSim Scripts, SC31-8945

WSim Script Guide and Reference, SC31-8946

WSim Utilities Guide, SC31-8947

WSim User Exits, SC31-8950

WSim Test Manager User's Guide and Reference, SC31-8949

Related publications

ACF/VTAM Programmer's Guide, SC23-0115

Systems Application Architecture Common Programming Interface Communications Reference, SC26-4399-06

Index

Printed in USA

SC31-8947-01