WSim Workload Simulator

# User's Guide

*Version 1 Release 1*

SC31-8948-01

WSim Workload Simulator

# User's Guide

*Version 1 Release 1*

SC31-8948-01

**Note!**

Before using this information and the product it supports, be sure to read the general information under"Notices" on page 163.

**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

# Contents

# Figures

# About this book

This book gives you the information you need to plan for Workload Simulator (WSim) tests and describes the requirements for installing and operating WSim on your system.

This book discusses the following topics:
- Installation and system requirements for using WSim
- Strategies for developing written test plans for tests
- The tasks you perform during tests. This information helps you decide which options, utilities, and features to use
- Running and controlling WSim
- Isolating problems encountered while running WSim.

This book enables you to assess what resources (computer, time, and human) you need to perform tests. It also helps you operate WSim by providing information on the WSim operator commands.

## Who should read this book

This book is intended for any WSim user who:
- Is responsible for developing test plans
- Wants a general overview of the tasks performed during tests
- Needs to install WSim
- Operates WSim daily.

Before using this book, you should be familiar with WSim and Systems Network Architecture (SNA) terminology and concepts.

## How to use this book

This book contains information from many of the books in the library for WSim and provides a comprehensive, task-oriented approach to planning. It also describes the requirements for installing WSim on your system. The sequence of topics in this book parallels the order of the tasks you perform as you plan for and conduct tests. This means that information about installing WSim is presented first, followed by information about determining the type of test and its objectives, configuring your system, defining the network and messages to be simulated, running the test, using WSim output, and operating WSim.

This book is designed to be the first book you use during testing. If you are a new WSim user, you should read the entire book and develop a test plan before you conduct your first test. Then, refer to individual chapters as needed to modify and refine your test plan. Pay special attention to sections that describe the planning considerations for using new or changed features. If you are responsible for installing WSim, concentrate on Installing WSim, which describes the installation and system requirements for WSim. Part 2. Operation provides information on the WSim operator commands.

Part 1. Planning and Installation, contains information on planning for and installing WSim.

- Chapter 1, “Getting started with WSim,” on page 3 provides an overview of WSim, describes the general process used in testing, and explains how you should use the other books in the WSim library.
- Chapter 2, “Installing WSim,” on page 7describes the installation and system requirements for using WSim.
- Chapter 3, “Testing with WSim,” on page 17 describes five types of tests and explains how you can benefit by testing with WSim.
- Chapter 4, “Creating a test plan,” on page 21presents a systematic approach to determining test objectives and to designing and writing a test plan for a test.
- Chapter 5, “Determining your system configuration,” on page 27 discusses how to configure the network you want to simulate and the real system used to run WSim.
- Chapter 6, “Defining the simulated network,” on page 37summarizes the planning considerations for coding the network definition statements that define the network to be simulated.
- Chapter 7, “Creating message generation decks,” on page 45 summarizes what you need to know to plan the message generation decks that generate messages for the simulated network.
- Chapter 8, “Running the test,” on page 61 explains the sequence for running tests and describes the various tools available for operation.
- Chapter 9, “Using WSim output,” on page 71 explains the tools and utilities available for analyzing the data produced by the test.
- Chapter 10, “Sample files,” on page 81 lists sample files provided with WSim.
- Chapter 11, “Using WSim to measure response times,” on page 83 describes features that enable WSim to measure system response times.
- Chapter 12, “Summary of logical unit (LU) types,” on page 85describes the logical units that WSim supports.

Part 2. Operation, provides details about operating WSim.

- Chapter 13, “Introduction to WSim operation,” on page 89 defines WSim and gives an overview of the information in this book.
- Chapter 14, “Running WSim,” on page 91provides information about the statements you need to operate WSim on MVS™ or TSO. This chapter also defines the execution parameters
- Chapter 15, “Using operator commands,” on page 99discusses console messages and operator commands and explains how to use operator commands when running on MVS or TSO.
- Chapter 16, “Using operator reports,” on page 117discusses the online reports printed by WSim.
- Chapter 17, “Controlling message logging,” on page 121 provides information about logging messages, including how to control, inhibit, and restart message logging.
- Chapter 18, “Using the Display Monitor Facility,” on page 125 discusses how to use the Display Monitor Facility to view simulated 3270 display images or non-3270 data streams on a display accessible from the VTAM* program.
- Chapter 19, “Isolating problems,” on page 133 discusses how to classify, isolate, and report problems.
- Chapter 20, “Specifying operator commands,” on page 137 lists the operator 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 173.

|  |  |
|---|---|
| **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. Planning and installation

# Chapter 1. Getting started with WSim

This chapter provides an overview of Version 1 Release 1 of Workload Simulator (WSim). It explains the general sequence you follow during a test and describes how you can use the other books in the WSim library during testing.

## What is Workload Simulator?

Workload Simulator (WSim) is a terminal and network simulation tool. You can use WSim to determine system performance and response time, to evaluate network design, 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, MVS/XA, or MVS/ESA (unless explicitly stated otherwise).

WSim enables you to test and evaluate teleprocessing systems without needing to have terminals and terminal operators present. You can use WSim to simulate the actions of a number of different applications and terminals. These simulated resources communicate with the real teleprocessing system (referred to as the system under test) as if they were physically present. No modifications to the system under test are required.

## How do you use WSim?

To use WSim, you write network definition statements that describe the network configuration to be simulated. You also create message generation decks that send and receive messages and enable your simulated terminals to take actions based on message content and network status. The network definition statements and one or more message generation decks form a script that WSim uses to send messages to the system under test. WSim then collects the information returned from the system under test, records that information for further analysis, and uses the information received to determine what to send to the system under test.

You can use WSim to perform tests that evaluate the reliability and approximate performance characteristics of a teleprocessing system under expected or projected operating conditions. That is, you use WSim to simulate a specified network of resources that generates a specified number and type of messages. WSim reduces the time and resources needed for testing and improves testing accuracy.

WSim enables the system under test to operate as it would under actual conditions. You can evaluate your teleprocessing system with a low volume of message traffic and proceed on an orderly basis to a high volume of traffic.

Because your programs and associated hardware are integral parts of the simulation process, WSim closely approximates a true operating environment.

WSim can generate the same or different messages for multiple terminals. This ability is especially useful in time-sharing environments. WSim can also simulate multiple networks concurrently to provide messages to drive multiple teleprocessing applications or multiple host processors.

### What can WSim simulate?

WSim can simulate the following types of resources:

- SNA logical units running as VTAM® application programs
- CPI-C transaction programs
- TPC/IP clients using Telnet 3270, 3270E, 5250, and Network Virtual Terminal (NVT), File Transfer Protocol (FTP), or simple TCP and UDP protocols attached to a TCP/IP network via the IBM TCP/IP for MVS product

## Conducting tests

The following sections discuss the general sequence you should use during testing and explain how you can use the other books in the WSim library during the test. The exact sequence you follow during testing depends on how familiar you are with WSim, the specific needs of your testing situation, and the operating procedures used in your organization.

Below is the general sequence of tasks you perform when conducting tests:

1. Install WSim.
2. Plan your test.
   a. Define test objectives.
   b. Design and write a test plan.
3. Configure your system.
4. Define the network to be simulated.
5. Create the message generation decks.
6. Run the test.
7. Use WSim output to analyze the results.

Note that planning continues throughout the test.

### Installing WSim

Installing WSim involves the following steps:

1. Understand the system and installation requirements. WSim is distributed on tape. The product tape installs WSim under MVS. Your system requirements vary depending on what system you are running.
2. Follow the specific installation instructions listed in the WSim Program Directory.
3. Authorize WSim to the operating system.

Refer to Chapter 2, “Installing WSim,” on page 7 for information about completing each of these tasks. The WSim Program Directory, which comes with the WSim distribution tape includes the most current information about installing WSim.

## Planning

Before you conduct a test, you need to define the objectives of your test and write a comprehensive test plan. This book explains what you need to create a test plan and describes the various utilities, options, and features that are available with WSim. Read Chapter 3, “Testing with WSim,” on page 17 to learn about the types of tests you can conduct with WSim. Then, refer to Chapter 4, “Creating a test plan,” on page 21 for more information about designing and writing a test plan for tests.

Planning is an extremely important part of testing. In many cases, the success or failure of a test hinges on how carefully it is planned in advance. You should view planning as an ongoing task and be prepared to refine your test plan until you obtain the wanted results. When beginning to use WSim, you should start with a small network definition and a simple message generation deck. For example, your first test might simulate a single terminal logging on to the operating system. Then, you can gradually add more terminals and use more complex message generation decks until you are simulating the complete network you plan to test.

## Configuring your system

Configuring your system involves determining what hardware and software resources you need and deciding how to arrange these resources. Since the resources for a test are both simulated and actual, you need to configure both the simulated network and the actual system. The configuration of the network that contains the resources you want to simulate (known as the logical configuration) determines the configuration of the system you use to run WSim (known as the physical configuration).

Chapter 5, “Determining your system configuration,” on page 27 presents four steps you can follow to configure your system. It also describes the types of logical and physical configurations that are used during tests.

## Defining the simulated network

After you determine the configuration of your system, you are ready to code the statements that define the configuration of your network. Chapter 6, “Defining the simulated network,” on page 37 discusses the general planning considerations for coding network definition statements. *Creating WSim Scripts* provides information about coding the statements for a particular simulation. The *WSim Script Guide and Reference* provides reference information about network definition statements and contains sample network definitions that can help you code your own definitions. Finally, if you want to perform additional processing of data and manipulation of network resources, refer to *WSim User Exits* for information about coding user exit routines.

## Creating message generation decks

After you define the network you want to simulate with WSim, you need to create message generation decks. Message generation decks define the messages that the simulated network sends to and receives from the system under test. They also enable the simulated network to take action based on the messages received.

The content of the message generation decks and the content of the network definition statements can be interdependent. Therefore, as you create the message generation decks, you may need to refer to and refine your network definition statements. You can simplify this task by planning for the interdependencies in advance. Refer to Chapter 6, “Defining the simulated network,” on page 37 and

Creating Message Generation Decks on page 45 for more information about the relationships between network definition statements and message generation decks.

Chapter 7, “Creating message generation decks,” on page 45 also provides specific planning information for using each of the following methods for creating message generation decks:

- The Structured Translator Language (STL) and the STL Translator. (For more information, refer to *WSim Script Guide and Reference.*)
- Message generation statements. (For more information, refer to *Creating WSim Scripts*.)
- Two script generating utilities:
  - WSim/IDC
  - ITPSGEN

For more information, refer to *WSim Utilities Guide*.

The *WSim Script Guide and Reference* provides reference information about the statements used in message generation decks and contains samples of message generation decks that can help you code your own message generation decks.

## Running the test

After you define the network to be simulated and created the message generation decks, you should run a sample test using a small configuration and a simple deck. This process is described in *WSim Scripts*. Then, after successfully completing one or more sample tests, you will be ready to run the complete simulation. Chapter 8, “Running the test,” on page 61 provides details about the planning considerations for running a sample test and for operating WSim. Refer to Part 2, “Operation,” on page 87for detailed information about starting, monitoring, controlling, and stopping simulations. Also refer to *WSim Messages and Codes* for explanations of the messages and return codes you may encounter during operation.

## Using WSim output to analyze the results

The final step in conducting a test is to use the WSim output to analyze the results. Chapter 9, “Using WSim output,” on page 71 provides planning information about the options available for generating and using output. Refer to *WSim Utilities Guide* and Part 2, “Operation,” on page 87 for information about how to use each of these options.

# Chapter 2. Installing WSim

This chapter provides an overview of things you need to do and questions you need to answer when installing WSim. It is intended to show you what is involved *before* you actually begin.

The step-by-step instructions you should follow once you are ready to install from tape can be found in the program directory shipped along with the tape.

For detailed information about the sample data sets included with the WSim installation tape, see Chapter 10, “Sample files,” on page 81.

After you install WSim, you can use ITPECHO to test the installation and to ease the learning and planning process. ITPECHO is VTAM application program supplied with WSim as a sample routine. Refer to *WSim Utilities Guide* for information about how to use ITPECHO.

## Understanding installation and system requirements

Requirements for installing WSim depend on the equipment you have and the resources you want WSim to simulate. The following sections organize the requirements for installing and running WSim into the following categories:

- MVS requirements
- General requirements

### MVS requirements

The following sections describe the requirements for installing and running WSim in an MVS environment. On MVS, WSim runs in virtual mode (V=V) or real memory mode (V=R).

#### Disk storage space requirements

On MVS, WSim requires disk storage space for partitioned data sets that contain the following:

- Network definition statements and message generation decks
- WSim host processor load modules
- Network definition statements
- Message generation decks
- Rate tables (optional) referenced by network definition statements
- Panels and CLISTs for the WSim/ISPF Interface
- EXECs, skeletons, and models for the WSim/ISPF Interface.

At least 1500 tracks of 3390 DASD are required to use WSim. If you plan to log messages to DASD, you may require more disk space for the WSim log data set. See Part 2, “Operation,” on page 87 for more information about allocating the log data set.

### Access method and authorization requirements

WSim uses QSAM (Queued Sequential Access Method), BSAM (Basic Sequential Access Method), or BPAM (Basic Partitioned Access Method) for tape, disk, and unit record operations. You can use any devices supported by these access methods for WSim data sets.

Authorization is optional when using the VTAM application interface to simulate terminals, but performance will be improved if authorization is provided. Such authorization enables WSim to use the VTAM authorized path facility. Refer to "Authorizing WSim on MVS" on page 10 for more information.

## General requirements

This section describes the general requirements for running WSim in an MVS environment. It lists the installation requirements for using the programs, interfaces, and facilities provided with WSim.

### Logging messages

To log the messages sent and received during a WSim simulation, you need at least one tape drive or disk data set. Refer toChapter 9, "Using WSim output," on page 71 and to *WSim Utilities Guide* for information about logging messages and formatting the log data set. Refer to Part 2, "Operation," on page 87 for information about using operator commands to control message logging.

### Printing output

To print output from the various WSim programs, you must have a printer with at least a 132-character print line.

### Using the VTAM Application Program Interface (API)

To use the VTAM Application Program Interface (API) to simulate SNA logical unit traffic, you must have any currently supported release of VTAM.

### Using the CPI-C transaction program support

To use the CPI-C transaction program support to simulate CPI-C client or server transaction programs (TPs), you must have VTAM Version 3 Release 2 or later.

### Using TCP/IP Telnet 3270, 3270E, 5250, or NVT, File Transfer Protocol (FTP), or simple TCP or UDP client support

To use the TCP/IP Telnet 3270, 3270E, 5250, NVT, FTP, or Simple TCP or UDP client support to simulate TCP/IP Telnet 3270, 3270E, 5250, NVT, FTP, or Simple TCP or UDP traffic requires the IBM TCP/IP product Version 2 Release 2 (or later) for MVS.

### Using the Display Monitor Facility

The Display Monitor Facility enables you to display simulated 3270 screen images on a VTAM-accessible monitor. To use the Display Monitor Facility, you must define it to WSim by using the DMAPPL execution parameter. You must define it to VTAM as an APPL in VTAM's VTAMLST. For a full description of these requirements, refer to Part 2, "Operation," on page 87

### Using the Interactive Data Capture Utility (IDC)

The Interactive Data Capture Utility (IDC) provides a way to capture data from a 3270 display communicating with a host application program and generate a script from the captured data. To use IDC, you must define it to VTAM as an APPL in VTAM's VTAMLST. Refer to *WSim Utilities Guide* for additional information.

The following utilities run in an MVS environment only:

### Using the WSim/ISPF Interface

The WSim/ISPF Interface is a panel-driven ISPF application that provides a user-friendly interface to most of the functions and utilities of WSim. To use the WSim/ISPF Interface, you must perform the installation steps as described in "Installing the WSim/ISPF Interface" on page 12.

## Communication controller requirements

To simulate network resources, you can operate WSim in the following physical configurations:

- VTAMAPPL for simulating VTAM primary and secondary LUs
- CPI-C transaction program for simulating CPI-C client or server transaction programs
- TCP/IP application for simulating Telnet 3270, 3270E, 5250, NVT, FTP, or Simple TCP or UDP clients

Chapter 5, "Determining your system configuration," on page 27 describes these physical configurations in detail and discusses the types of network resources you can simulate with them.

### VTAMAPPL configuration

In the VTAMAPPL configuration, WSim can simulate primary and secondary logical units (LUs) in the same subarea as VTAM. However, you can also use the VTAMAPPL configuration to send and receive messages from simulated resources in different domains.

### CPI-C transaction program (TP) configuration

In the CPI-C TP configuration, WSim can simulate client or server transaction programs in the same subarea as VTAM. However, you can also use the CPI-C TP configuration to send and receive messages from simulated TPs in different domains.

### TCP/IP application configuration

In the TCP/IP application configuration, WSim can simulate Telnet 3270, 3270E, 5250, NVT, or FTP clients communicating with Telnet 3270, 3270E, 5250, NVT, or FTP servers, such as TCP/IP for MVS. WSim can also simulate Simple TCP or UDP clients communicating with various types of servers.

## Printing the WSim program directory

WSim has a program directory shipped with the MVS product distribution tape for installation on MVS. The program directory contains detailed instructions you need to install and maintain WSim.

In an MVS environment, you can use the JCL shown below to print a copy of the WSim Program Directory from the MVS product distribution tape.

```
//PRPGMDIR JOB
//*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
//*     SAMPLE JCL TO PRINT WSim PROGRAM DIRECTORY FROM TAPE  *
//*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
//COPY     EXEC PGM=IEBCOPY
//SYSPRINT DD  SYSOUT=A
//TAPE     DD  DSN=IBM.H281110.F3,DISP=(OLD,PASS),UNIT=TAPE,
//             LABEL=(4,SL),VOL=SER=281110
//DISK     DD  DSN=&&TEMPDS,DISP=(NEW,PASS),UNIT=SYSDA,
//             SPACE=(80,(5000,5000,1))
//SYSIN    DD  *
```

```
 COPY INDD=TAPE,OUTDD=DISK
 SELECT MEMBER=PGMDIR
/*

//PRINT    EXEC PGM=IEBPTPCH
//SYSPRINT DD  SYSOUT=A
//SYSUT1   DD  DSN=&&TEMPDS(PGMDIR),DISP=(OLD,DELETE)
//SYSUT2   DD  SYSOUT=A
//SYSIN    DD  *
 PRINT PREFORM=M
/*
```

The WSim Program Directory is printed on the printer designated by the SYSPRINT statement. For example, the following statement designates a printer with SYSOUT class A:

```
//SYSPRINT DD SYSOUT=A
```

## Authorizing WSim

The following sections describe how to authorize WSim on MVS and TSO.

### Authorizing WSim on MVS

WSim executes more efficiently in the VTAMAPPL and CPI-C environments if it is authorized using the requirements of the authorized program facility (APF). You may have to follow several steps to provide system authorization for WSim. For details, check the planning and user documentation for the system on which WSim operates. Refer to the *MVS/ESA System Program Library: Initialization and Tuning Guide* for program authorization information for MVS.

To be authorized, the WSim load modules must reside in an installation-defined authorized library or in a private library that you create and authorize, such as WSIM.SITPLOAD. To authorize a user library, you must add its name to the IEAAPF00 member of SYS1.PARMLIB or to the corresponding APF data set indicated in the MVS initialization parameters. The following list shows sample entries for the IEAAPF *xx* member in SYS1.PARMLIB that are required to authorize WSim execution on MVS.

| Entry | Explanation |
|---|---|
| **WSIM.SITPLOAD** *yourpack* | Authorizes the WSim load module. |

If WSim runs in a heavily loaded host processor, you may want to mark the WSim load module ITPENTER as nonswappable. Add ITPENTER to the list of programs in the program properties table (PPT) and set the correct entry flags. Using *MVS/ESA Initialization and Tuning Reference* (GC28-1635) as a guide, code the following in your SCHED *nn* member:

```
PPT   PGMNAME(ITPENTER) NOSWAP
```

### Authorizing WSim under TSO

You take two different steps when you authorize WSim to run under TSO. The following list briefly describes these steps; refer to *OS/VS2 System Programming Library: TSO/E Customization* for complete details.

1. If you are installing WSim under TSO/E and the member IKJTSO00 exists in SYS1.PARMLIB, add the WSim load module ITPENTER to the list of authorized programs contained in the member IKJTSO00. There are two places where you need to do this, as shown in the example below.

```
.
.
.
               /*                         */
AUTHPGM NAMES(     /* AUTHORIZED PROGRAMS     */     +
   IEBCOPY         /*                         */     +
   ICHDSM00        /* RACF PROGRAMS           */     +
   ICHUT100        /*                         */     +
   ICHUT200        /*                         */     +
   ICHUT400        /*                         */     +
   ITPENTER        /*                         */     +
   JMS             /*                         */     +
           )       /*                         */
                   /*                         */
   .
   .
   .
                   /*                         */
AUTHTSF NAMES(     /*PROGRAMS TO BE AUTHORIZED*/     +
                   /*WHEN CALLED THROUGH THE  */     +
                   /*TSO SERVICE FACILITY.    */     +
    IEBCOPY        /*                         */     +
    JMS            /*                         */     +
    ITPENTER       /*                         */     +
    IKJEFF76)      /*                         */
                   /*                         */
```

2. If IKJTSO00 does not exist, add the WSim load module ITPENTER to the list of authorized programs in the control section (CSECT) IKJEFTE8 of the module IKJEFT02.

   **Note:** TSO/E uses the IKJTABLS module instead of IKJEFT02. The CSECT is still IKJEFTE8.

   You can do this in one of the following ways:

   - Zap CSECT IKJEFTE8 of module IKJEFT02 and add the EBCDIC characters for ITPENTER in the next unused entry.
   - Change the source code for CSECT IKJEFTE8 and relink this module.
   - Use an SMP/E USERMOD.

3. List the following WSim data set as authorized in the IEAAPF00 member of SYS1.PARMLIB or the corresponding APF data set indicated in the MVS initialization parameter.

   **WSIM.SITPLOAD** The load module for WSim.

**Notes:**

- When WSim runs as an authorized program, all libraries from which it runs other programs (such as your libraries containing your user exit modules) must be accessible and must themselves be APF authorized. If you specify these libraries in the STEPLIB DD statement of your TSO logon procedure, all other libraries you specify on that STEPLIB must also be authorized. This includes the ISPF libraries if you specify them there.

  Another way you can make the libraries available to WSim is to include them in the LNKLST00 member of SYS1.PARMLIB, thus causing them to be concatenated to SYS1.LINKLIB and always available. If you do this, you need not specify the libraries on the STEPLIB DD statement and libraries you specify there (such as ISPF libraries) do not need to be authorized.

  You can do other combinations, as long as the libraries from which WSim runs programs are authorized and are still considered authorized when you run WSim.

- If you will be using WSim only for VTAM application simulation, you do not need to authorize WSim on MVS, but WSim will execute more efficiently in the VTAMAPPL environment if it is authorized.

## Installing the WSim/ISPF Interface

After you install WSim, you must do some additional setup to run the WSim/ISPF Interface. Follow the steps below to install the WSim/ISPF Interface.

1. Concatenate the following WSim data sets into your TSO logon procedure: Add the following data sets to your TSO logon procedure:

| | |
|---|---|
| **WSIM.SITPPNL** | Concatenate this data set to the ISPPLIB DD. |
| **WSIM.SITPMSG** | Concatenate this data set to the ISPMLIB DD. |
| **WSIM.SITPEXEC** | Concatenate this data set to the SYSEXEC DD statement. |
| **WSIM.SITPTBL** | Concatenate this data set to the ISPTLIB DD statement. |
| **WSIM.SITPSKEL** | Concatenate this data set to the ISPSLIB DD statement. |

2. Connect the WSim/ISPF Interface to your ISPF/PDF system.

   To do this, you must tailor your ISPF/PDF Primary Option Menu to include an option to invoke the WSim/ISPF Interface. Contact your system programmer to locate the source of this menu. A sample menu appears in “Installing the WSim/ISPF Interface.”

```
%----------------------- ISPF/PDF PRIMARY OPTION MENU ------------------------
%OPTION ===>_ZCMD                                                             +
%                                                            +USERID - &ZUSER
% 0 +ISPF PARMS   - Specify terminal and user parameters    +TIME   - &ZTIME
% 1 +BROWSE       - Display source data or output listings
% 2 +EDIT         - Create or change source data
% 3 +UTILITIES    - Perform utility functions
% 4 +FOREGROUND   - Invoke foreground language processors
% 5 +BATCH        - Submit job for language processing
% 6 +COMMAND      - Enter TSO command or CLIST
% 7 +DIALOG TEST  - Perform dialog testing
% F +WSIM/ISPF    - Invoke WSim/ISPF Interface
% T +TUTORIAL     - Display information about ISPF/PDF
% X +EXIT         - Terminate ISPF using log and list defaults
%
+Enter%END+command to terminate ISPF.
%
)INIT
  .HELP = ISR00003
  &ZPRIM = YES        /* ALWAYS A PRIMARY OPTION MENU */
  &ZHTOP = ISR00003   /* TUTORIAL TABLE OF CONTENTS   */
  &ZHINDEX = ISR91000 /* TUTORIAL INDEX - 1ST PAGE    */
)PROC
  &ZSEL = TRANS( TRUNC (&ZCMD,'.')
              0,'PANEL(ISPOPTA)'
              1,'PGM(ISRBRO)'
              2,'PGM(ISREDIT)'
              3,'PANEL(ISRUTIL)'
              4,'PANEL(ISRFPA)'
              5,'PGM(ISRJB1) PARM(ISRJPA) NOCHECK'
              6,'PGM(ISRPTC)'
              7,'PGM(ISRYXDR) NOCHECK'
              F,'CMD(ITP0MAIN prefix)' /* Entry for WSim/ISPF Interface */
              T,'PGM(ISPTUTOR) PARM(ISR00000)'
            ' ',' '
              X,'EXIT'
              *,'?' )
   &ZTRAIL = .TRAIL
)END
```

where *prefix* is the initial qualifiers for the WSim/ISPF Interface data set. Refer to *WSim Utilities Guide* for more information.

*Figure 1. Example of updated ISPF/PDF primary option menu*

3. Allocate and catalog any data sets you may be using. A WSIM EXEC (ITP0INST) is provided that allocates all the data sets required by the WSim/ISPF Interface. This allocates the default data sets for your system and sets up system defaults for your installation qualifier.

   **Note:** Only run this program once. If you already ran this program and it failed while allocating your data sets, delete any data sets the program allocated and run this program again.

   Modify this EXEC when you install the WSim/ISPF Interface to set up global default values for all users of your system and also to adapt to your system. Places where you should make changes are highlighted with a comment block stating that you can or should make a change there.

   If you do not modify or run this exec, the WSim/ISPF Interface will still run properly. However, each user of your system may need to set certain default values from within the WSim/ISPF Interface that this exec defines.

   This EXEC creates a data set on your system named *prefix*.ITP0INST.SETUPXXX. *prefix* is the initial qualifier used in defining this data set. The default is WSIM$$$$. The last 3 characters of the name denote a version/release identifier and a $ sign. Initially, this value is 11$. This data set lets users run multiple releases of WSim concurrently. Put this data set on a publicly accessible disk and do not modify this data set.

If this data set exists when you run the WSim/ISPF Interface the first time for the current release, it uses the values within the data set as defaults. Otherwise, it uses its own defaults. Specifying the PROFILE execution parameter when you run the WSim/ISPF Interface also uses the values in the profile data set, whether this is the first time or not.

The exec ITP0INST allocates the data sets shown in Table 1 with their associated parameters. You can change the exec, use existing data sets, or allocate your own data sets using option 3.2 of ISPF/PDF.

*Table 1. WSim/ISPF Interface data set allocations*

| Description | Data Set Name | Attributes |
|---|---|---|
| WSim Networks (INITDD) | WSIM.TESTFILE | PDS, LRECL=80, SPACE=(8800,(50,25,8)), UNIT=SYSALLDA, RECFM=FB |
| WSim Message Decks (MSGDD) | WSIM.MSGFILE | PDS, LRECL=80, SPACE=(8800,(150,50,8)), UNIT=SYSALLDA, RECFM=FB |
| STL Input Data Sets | WSIM.STLIN | PDS, LRECL=255, SPACE=(8800,(250,100,10)), UNIT=SYSALLDA, RECFM=VB |
| WSim Scripts | WSIM.NETWORK | PDS, LRECL=80, SPACE=(8800,(150,50,8)), UNIT=SYSALLDA, RECFM=FB |
| WSim Log Data Set | WSIM.LOGDATA | SEQ, LRECL=8188, SPACE=(8192,(400,400)), UNIT=SYSALLDA, RECFM=VB, DCB=(NCB=5) |
| WSim Control Statements and Commands | WSIM.CONTROL | PDS, LRECL=80, SPACE=(8800,(5,3,1)), UNIT=SYSALLDA, RECFM=FB |
| WSim IDC Defaults | WSIM.IDCDFLTS | SEQ, LRECL=2087, SPACE=(2087,(1,0)), UNIT=SYSALLDA, RECFM=FB |

4. Update the load library name and models qualifier on the SETUP panel in the WSim/ISPF Interface. If you specified these names as WSIM or if you specified this in ITP0INST, you need not change them on the SETUP panel.
5. Authorize WSim to run under TSO if you plan on running your WSim simulations interactively. See "Authorizing WSim under TSO" on page 10 for instructions on authorizing WSim to run under TSO.

## WSim library setup

After you install WSim, you have the following data sets available. The names shown below are the default names shipped with the product. The names your installation uses may differ from the ones listed. Contact your system programmer if you are unsure of these names.

| | |
|---|---|
| **WSIM.SITPLOAD** | Contains the WSim load library modules. |
| **WSIM.SITPPNL** | Contains all the panels used by WSim. |
| **WSIM.SITPMSG** | Contains all the panel error messages used by WSim. |
| **WSIM.SITPEXEC** | Contains all the REXX EXECs used by WSim. |
| **WSIM.SITPTBL** | Contains all the ISPF tables used by WSim. |
| **WSIM.SITPSKEL** | Contains skeleton files used by WSim for building batch jobs. |
| **WSIM.SITPMDLS** | Contains the model scripts used when creating WSim STL input. |
| **WSIM.SITPMDLM** | Contains the model scripts used when creating scripts (networks and message generation decks). |

## Checklist for installing WSim

The following checklist summarizes the steps you follow when installing WSim and indicates where you can find instructions for performing each step.

1. Plan for installation and system requirements. For more information, see "Understanding installation and system requirements" on page 7.
2. Install WSim. See WSim *Program Directory* for specific installation instructions.
3. Plan for authorization requirements. For more information, see "Authorizing WSim" on page 10.
4. If you are going to run WSim using ISPF, install the WSim/ISPF Interface. See "Installing the WSim/ISPF Interface" on page 12 for more information.
5. To verify WSim installation, run the sample script named INSTALL1.See *Creating WSim Scripts* for a listing of this script.
6. Use the Loglist Utility to analyze the output from running the sample script. For information about using the Loglist Utility, see *WSim Utilities Guide*.

# Chapter 3. Testing with WSim

This chapter describes five types of tests that you can conduct with WSim and discusses the types of changes you should test. It concludes with information of the costs of testing. Read this chapter to learn about how you can use WSim in your organization.

## What can you test?

The first step in using WSim is to understand what types of tests you can conduct. Even if you do not use WSim for all these tests, regard them as starting points for your plans. You can use WSim for the following types of tests:

- Function
- Regression
- Performance
- Stress
- Capacity planning.

The following sections discuss each of these types of tests.

### Function tests

Use function tests to test a function of your system independently of other functions. This type of test answers the question Does it work right? Some of the functions that you can test with WSim include new application transactions, logon and logoff sequences, error transactions, new hardware attachments, and new software products. For example, you would use a function test to determine if each function of your new application program works as designed.

A significant advantage to using WSim for function tests is that you can save the scripts and use them for later regression or stress tests.

### Regression tests

Regression tests verify that old functions still operate correctly after you add new functions or made other changes to the system. You can use this type of test to answer the question Does it *still* work right? For example, you could use a regression test to determine if you introduce errors into your network by installing the latest version of your application or by changing your network configuration.

There are many advantages to using WSim in a regression test:

- Scripts are repeatable. After you create a script, you can save it in a testcase library for future use.
- You can use scripts to check for errors in system responses. You can define logic tests to compare the actual response against the expected response. If there is a discrepancy, the script can handle the error or write error messages to the operator console or the log data set.
- WSim can run automatically. After you start the WSim job, execution parameters and operator commands in the scripts can control WSim operation. A script can even end the simulation when all the test cases are complete.

- You can use the Log Compare Utility to compare the 3270 display records from two log data sets. This comparison enables you to determine how a change to the system affects the behavior of IBM 3270 Information Display Systems without having to compare the log data sets manually.

You do not need to restrict WSim to large comprehensive regression tests. You can easily use WSim with existing test cases to test routine maintenance changes to your system.

## Performance tests

Performance tests consist of measurement and tuning. When you conduct a performance test, you measure system performance, tune the system, and then measure again. You can use this type of test to answer the question How well can I make it work? For example, you would use a performance test to determine how a change in region size would affect the performance of your system. To conduct successful performance tests, you must be able to do the following:

- Measure system performance (for example, throughput or host processor utilization) at a particular transaction rate and load and with certain system parameters.
- Change the system parameters.
- Measure system performance again at the same transaction rate and load.

For example, you might measure throughput or host processor utilization before and after changing buffer sizes or priorities.

There are two major advantages to using WSim for performance tests. First, you can use WSim to obtain a controlled, repeatable transaction load on the system. Second, you can use WSim to report terminal response times.

If you are conducting a performance test, you should run WSim in a different host than you are using for the system under test. Otherwise, the computer resources needed for WSim operation can impact your results.

## Stress tests

Stress tests try to find problems in interactions and resource contentions by driving the system to extremely high transaction rates. These types of problems might be missed by single terminal simulations. You can use stress tests to answer the question What will break first? For example, you might use a stress test to determine how many users can log on to your system before it is overloaded.

It is nearly impossible to conduct a stress test on an online system without a tool like WSim. Using WSim, you can generate controlled message traffic at controlled rates without worrying about coordinating terminal operators or risking a possible system crash.

If you are conducting a stress test, you should run WSim in a different host than you are using for the system under test. Otherwise, the computer resources needed for WSim operation can impact your results.

## Capacity planning tests

Capacity planning tests predict how your system behaves when new resources are brought online. These tests can also intentionally overuse one or more system resources to determine if the system can perform adequately with a predicted increased workload. You can use capacity planning tests to answer the question

What will happen if I add this many resources to my system? For example, you would use capacity planning tests to predict whether your system will perform adequately if you add 20 new terminals.

WSim enables you to drive a system with a higher transaction rate than normal without the need for terminal operators. You can also verify previous capacity projections with a simulation. In addition, some types of capacity planning tests are impossible to conduct without using a testing tool such as WSim. WSim enables you to simulate more terminals or different types of terminals than you have in your production network.

## What should you test?

In general, you should use WSim to test any significant change to your system. This includes any changes that might affect hardware, system software, application programs, workload, or environment. Ask yourself the following questions about a proposed change:

- How complex is the change?
- How many components will be changed?
- How many transactions or transaction types will be affected by the change?
- How many people will be affected by the change?
- How difficult will backup and recovery be after the change?
- What's my past experience with changes of this type?

The answers to these questions help you decide what types of changes you want to test with WSim.

## What is the cost of testing?

In general, the cost of testing is less than the cost of not testing.

The cost of testing includes the expense of test personnel who could be doing other work, test hardware, and test tools (for example, a WSim license).

## What is the cost of not testing?

The costs of not testing or of testing inadequately may include the following:

- More system outages
- More personnel for production system maintenance
- Slower software migration
- End-user dissatisfaction and potential loss of business
- Lost data
- Less new development
- Higher data processing costs
- Missed business opportunities.

The main objective of testing online systems is to find problems before you put the system into production. It is much less expensive to fix a problem found during testing than it is to have a system fail during production and then have terminal operators sitting idle.

There is no shortcut to the comprehensive testing of an online system. If your organization requires a high degree of system reliability, high availability, and good system performance, then you must dedicate significant effort and resources to testing.

The resources you need to run a test depend on what type of test you plan to conduct and how complex your system is. In general, you should treat the setting up of a test system like an application development program. You need to dedicate people to designing and developing the test system, but after it is developed you can often coordinate testing with only one or two people. The amount of time you need to code the test cases depends on how many different transaction types you have and how complex each transaction is.

However, as with any application development program, you can reduce the cost of testing by carefully planning your test in advance and by providing the test developers with a comprehensive test plan. For more information about creating a test plan for tests, refer to Chapter 4, “Creating a test plan,” on page 21.

# Chapter 4. Creating a test plan

This chapter presents a systematic approach to creating a test plan for your test. It emphasizes the importance of establishing test objectives *before* you plan for a particular test. Then, it discusses what you need to design a test plan for WSim. Finally, it describes one way you can organize the information in your test plan.

## Establishing test objectives

Before you can plan for or conduct a test, you need to establish objectives for the test. You should consider why you want to use WSim and what you are trying to test. You should also consider what results you are trying to achieve. Finally, you should state your objectives in a clear and concise paragraph or list. Clearly stating your objectives before you begin planning makes the planning process easier and enables you to write a more comprehensive and useful test plan. Your statement of objectives should have two parts:

- Purpose of the test
- Expected results.

These parts are discussed in the following sections.

### Purpose

The first part of your statement of objectives describes the purpose of the test. For example, suppose that you want to determine how well you can make your MVS system work under conditions of typical use. You want to adjust system parameters and test the system to find the combination of parameters that optimize performance. To do this, you decide to use WSim to simulate 60 users logging on to TSO from different domains in your network. You can adjust system parameters such as buffer sizes and transmission priorities and repeat the tests until performance is optimized.

Your statement of purpose should identify the type of test you are planning to conduct. The type of test can determine where you run WSim. For example, if you want to conduct a performance or stress test, you should run WSim in a separate host processor to ensure that WSim operation does not affect the performance of the system under test. Refer to Chapter 3, “Testing with WSim,” on page 17 for a complete discussion of the types of tests and their uses and requirements.

### Expected results

The second part of your statement of objectives describes the results you expect. You should try to state the expected results in measurable terms; for example, The MVS system must be able to handle 60 logon requests in 5 minutes with 95% of the requests satisfied in 10 seconds each.

## Designing a test plan

After you have stated the objectives for your test, you are ready to begin designing the test plan. The test plan documents the test you want to conduct.

A test plan outlines the steps to follow during the test and provides information for conducting the test. The form of the test plan and the amount of detail it includes depend on what you are testing, the special requirements of your test situation, and the procedures used in your organization. Not surprisingly, a test plan for using WSim to test a new application looks different from a test plan for testing new network resources. Similarly, a test plan for a very complex regression test looks different from a test plan for a simple function test.

In general, the more detailed and explicit your test plan is, the easier it will be to conduct the test. Detailed test plans enable someone who does not know much about your system or WSim to create the script and conduct the test, whereas less detailed test plans require a high degree of familiarity with your system and WSim. Detailed test plans also provide a valuable record of what you did during a specific test. This record will help you analyze your results and plan future tests.

Before you can write the test plan, you need to decide what types of information it will include and how you should organize it. To do this, you need to answer a number of questions. Using your statement of objectives as a guide, answer the following questions by reading the other chapters in this book and by referring to the other books in the WSim library:
- What types of resources do you want to simulate?
- What actual resources do you need to conduct the test?
- What types of transactions do you want to test?
- How do you want to run WSim?
- What kind of output do you want to obtain from WSim and how do you expect to use this output?

The answers to these questions depend on your test objectives and they determine how you organize and write the test plan. For example, if you are testing new network resources, you should list specific resources that you want to simulate. However, you may not need to specify the types of messages that you want these simulated resources to send. In contrast, if you are testing a new application, the test plan should include a detailed description of the transactions to be tested.

## Writing a test plan

After you have answered these questions, you can begin writing your test plan. The following sections present one way to organize a test plan for WSim. They discuss the general types of information you can include in your test plan. Depending on your situation, you may need to include additional information.

A test plan for a test can include the following types of information:
- Introductory and background material
- Statement of objectives
- Resources needed including:
  - Hardware and software
  - People
  - Time
- Specifications for the following steps:
  - Defining the resources to be simulated
  - Creating the message generation decks
  - Running WSim
  - Using WSim output
- Test procedures, including:

- – Entry and exit criteria
- – Procedures for conducting sample tests
- – Procedures for reporting progress and status
- – Schedule.

Regardless of how you organize this information, your test plan should include as much detail as you need to conduct a successful test. As you gain experience using WSim, you will find the test plan format that works best for your situation and your organization. The following sections discuss the various parts of a test plan.

## Introduction

Use the introduction to give a high-level overview of what you are testing, why you are testing it, and who is doing the testing. This section should mention relevant background information, such as the results of any previous tests. It should also briefly discuss problems leading to the need for testing, plans for future tests, and so on.

## Objectives

Use the objectives section to list the objectives for the test. As discussed in "Establishing test objectives" on page 21, you should write a complete statement of objectives for the test before you design or write the test plan. By stating your objectives in the test plan, you ensure that everyone involved in the test knows its purpose and expected results.

In addition to stating the purpose of the test and the expected results, the objectives section should specify what to do if the expected results are not achieved. For example, if your MVS system is unable to satisfy 60 logon requests in 10 seconds each, you might do one of two things:

- Change your objectives
- Attempt to fix the problem and retest the system.

## Resources

The resources section of the test plan lists the resources you will use during the test. The resources for a test include the hardware, software, time, and people needed to conduct the test.

### Hardware and software

The first part of the resources section lists the hardware and software you need to run the test. This includes resources needed to run WSim as well as the actual resources in the system under test. The types of resources that you would list in this section include the following:

- Host processors
- Operating systems
- WSim
- Other telecommunications software such as VTAM or NCP
- Software applications you will use
- Data bases or data sets you will access
- Other monitoring and testing programs.

You should specify the model or release level of each resource as well as any other information that would be helpful.

### Time

The second part of the resources section lists the time needed to complete the test. This includes the time needed to plan for and conduct the test as well as the time needed to analyze and report the results. If you are using WSim for the first time, the time section can also include an estimate of the time needed to install WSim, generate the control programs, and learn to use WSim.

For an example schedule for a test, refer to “Schedule” on page 25.

### Staffing

The third part of the resources section lists the people needed to conduct the test. These include the people needed to install WSim (if necessary), define the network, create the message generation decks, run WSim, and analyze the results. You may not need different people for each of these steps, but you should consider each of these tasks as you plan your staffing needs.

## Test specifications

Use the test specifications section of the test plan to describe the specifications for your test. You can divide this section into the following parts:

- Specifications for defining the network that you want to simulate. This part lists the resources you want to simulate and may include devices, logical units, applications, and so on. The amount of detail depends on your situation.
- Specifications for creating the message generation decks. This part describes the transactions that you want to test and includes any other information needed to create the message generation decks. Again, the amount and type of detail varies with each test.
- Specifications for operating WSim. This part lists the utilities, features, and options that you want to use when you run the test.
- Specifications for WSim output. This part lists the types of output that you want to get from the test and how you will get them.

## Testing procedures

The procedures section describes the general procedures you will use to conduct the test. It includes information about the following topics:

- Entry and exit criteria
- Sequence of sample tests
- Procedures for reporting problems and status
- Schedule.

### Entry and exit criteria

The entry and exit criteria for the test are closely related to the purpose and expected results for the test. The ***entry criteria*** are the conditions that must be met before you can start the test. For a simple test, this section of the test plan would list the hardware and software that needs to be operational. For a more complicated test involving a new application, this section might specify at what stage of development the application needs to be.

This section also describes the ***exit criteria***, the conditions that must be met before the test is completed. Exit criteria apply more to function tests than to other types of tests. For example, the exit criteria for a function test on a new application might require that all problems be resolved with a permanent fix and that all testcases be run successfully with the fixes in place.

## Sequence of sample tests

As explained in “Running and analyzing a sample test” on page 61, you should divide the test into a number of sample tests. These sample tests enable you to test portions of the script before you run the complete simulation. You can use the test procedure part of the test plan to describe the sequence of sample tests.

## Procedures for reporting problems and status

This part of the testing procedures section describes the procedures that should be followed to report problems with the system under test, the status of the test, and the results of the test. By establishing these procedures in advance, you can ensure that everyone involved in testing knows what kinds of information to be aware of and what to do when problems arise. This section should include a procedure for documenting the following:

- Problems with the performance of the system under test
  - What the problem is
  - Who found it
  - Where and how it was found
  - When it was found
  - How and when it was resolved
  - Who fixed it
  - Who checked the fix and when
- Status of the test
  - What the actual progress is compared to the scheduled progress
  - What problems are being encountered
- Results of the test
  - What the actual results are
  - How they compare to the expected results.

## Schedule

The final section of the test plan indicates the schedule to follow during the test. You should indicate when you plan to begin the test and how long you expect it to take. You may want to include target dates by which you will complete each sample test. “Schedule” shows an example schedule for a test that will simulate 60 terminals logging on to TSO.

*Figure 2. Example schedule for a test*

| TASK | START DATE |
|---|---|
| Start planning | 6/01 |
| Test plan completed | 7/01 |
| Run sample test 1 (one terminal logging on to TSO) | 8/01 |
| Run sample test 2 (several terminals logging on) | 8/08 |
| Run sample test 3 (60 terminals logging on) | 8/22 |
| Run sample test 4 (1 terminal, complicated logon script) | 8/29 |
| Run complete simulation (60 terminals logging on, complicated script) | 9/06 |
| Analyze results and retest (as needed) | 9/20 |
| Report results | 10/03 |

Note that the amount of time it takes to conduct a test depends on how complex your test is as well as on how familiar you are with WSim. If you are using for the first time, you should schedule additional time for installation and learning.

# Chapter 5. Determining your system configuration

This chapter discusses how to configure your system for a test. When you configure your system, you determine what hardware and software resources you need and decide how to arrange them. Once you have configured your system, you can begin coding the network definition statements and the message generation decks for your test.

Before you read this chapter, be sure that you are familiar with the definitions of the following SNA terms and WSim concepts:

**Logical unit (LU)**
A logical unit is a port through which an end user accesses the SNA network to communicate with another end user or the system services control point (SSCP).

**Transaction program (TP)**
In WSim, a transaction program is any program that uses LU type 6.2 communication protocols to communicate with another program. WSim implements transaction programs using the Common Programming Interface for Communications (CPI-C).

**Session**
A session is a logical connection that enables two network addressable units to communicate with each other (for example, an LU-LU session or an SSCP-LU session). Each half of a session is a ***half-session***.

## Understanding logical and physical configurations

Before you can conduct a test, you need to configure both the network you want to simulate and the system you will use to run WSim. The configuration of the network that contains the resources you want to simulate and the real system you are testing is called the ***logical configuration***. Depending on what you want to simulate, you can use one of several types of logical configurations when you conduct a test. For each logical configuration, you must use a specific ***physical configuration***, which is the configuration of the system you use to run WSim. This chapter describes logical and physical configurations in more detail.

### Logical configuration

The logical configuration describes how a network that contains the resources you want to simulate is arranged. It includes the resources that are simulated by WSim as well as the actual resources you are trying to test. These actual resources make up the ***system under test***.

For example, if you want to conduct a stress test on your MVS system, the simulated resources might consist of a number of terminals. If you want to simulate users accessing a new VTAM application, the actual resource (system under test) consists of a real host computer running VTAM and the VTAM applications.

## Physical configuration

The physical configuration describes how the actual hardware and software resources needed for a particular test are arranged. These resources include the resources needed to run WSim and the resources in the system under test. They can include the following:

- Host processors
- System software, including
  - Operating system
  - Access method (for example, VTAM)
- Application software
- Test monitoring programs
- WSim.

The next section of this chapter presents an example that demonstrates how to configure your system for a test. The final section of this chapter provides more detail about each of the logical and physical configurations you can use during tests.

## Configuring your system

You can use these steps to configure your system for WSim:

1. Determine the objectives for your test
2. Determine what resources you want to simulate
3. Decide how a network containing these resources would be arranged (the logical configuration)
4. Identify the physical configuration.

After you have determined the logical configuration, you can identify the physical configuration you need to conduct the test. The physical configuration includes the resources in the system under test as well as the resources needed to operate WSim.

You can operate WSim in any of the following physical configurations:

**VTAMAPPL configuration**
You use this configuration to simulate primary and secondary logical units in the same subarea as VTAM. These logical units can have a session with any other logical unit that VTAM will allow a session to be started with.

**CPI-C transaction program configuration**
You use this configuration to simulate client and server CPI-C transaction programs (TPs) in the same subarea as VTAM. These TPs can have a conversation with any other TP on any LU to which VTAM will allow a conversation to be started.

**TCP/IP Application Configuration**
You use this configuration to simulate Telnet 3270, 3270E, 5250, NVT, or FTP clients. These simulated clients can have a session with any Telnet 3270, 3270E, 5250, NVT, or FTP server that TCP/IP allows. This configuration can also be used to simulate Simple TCP or UDP clients in session with various servers.

## Summary

You can use this procedure regardless of what you want to simulate. However, the logical and physical configurations you use depend on what you are simulating. The next section of this chapter describes the three logical and physical configurations that you can use for tests in more detail.

## Determining logical and physical configurations

Depending on what you want to simulate, you can use one of the following types of logical configurations in a test:

- VTAM application simulation
- CPI-C transaction program simulation
- TCP/IP application simulation

The following sections of this chapter discuss these logical configuration types and the corresponding physical configurations required to simulate them. Because you will probably use only one of these configurations for most of your tests, you can concentrate on those sections that apply to your specific situation. However, you will probably want to read the other sections to learn about other ways you can use WSim.

### VTAM application simulation

Figure 3 illustrates the logical configuration of a network in which you want to simulate SNA logical units that are accessing a VTAM application. These logical units could represent terminals or other VTAM applications. You would use this logical configuration to test VTAM applications or subsystems such as IMS/VS, Customer Information Control System/Virtual Storage (CICS*/VS) or TSO. This VTAM subarea could be part of a large network with many other nodes.

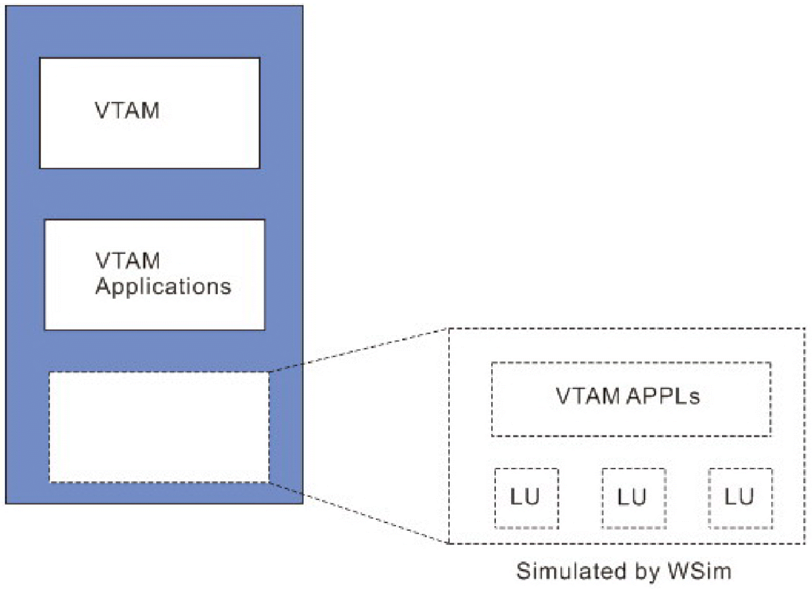


*Figure 3. Logical configuration for VTAM application simulation*

The physical configuration that corresponds to this logical configuration is the VTAM application (VTAMAPPL) configuration, which is shown in Figure 4.

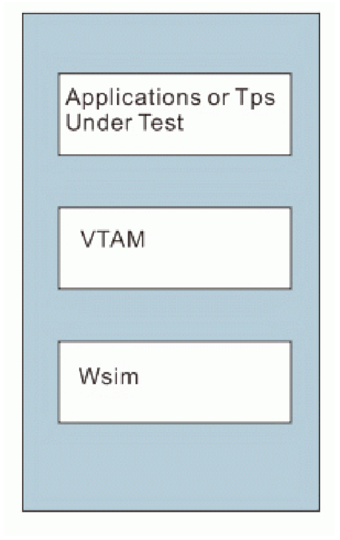


*Figure 4. Physical configuration for simulation of a VTAM application or CPI-C transaction program*

In the VTAMAPPL configuration, WSim runs as a VTAM application program and uses the standard VTAM Application Program Interface (API). Because VTAM treats an application program as a logical unit, WSim can simulate logical unit half-sessions that look like SNA terminals to other VTAM application programs.

WSim does not drive any hardware directly when running exclusively as a VTAM application. Instead, it uses the VTAM API to send and receive messages, eliminating the need for any extra hardware resources. To run as a VTAM application program without a communication controller, WSim requires only a currently supported release of VTAM and the VTAMLST APPL definitions in VTAM that enable WSim to communicate with VTAM.

WSim only simulates local SNA logical units within a VTAM subarea. However, this does not limit a simulated logical unit to same-domain sessions since VTAM may be able to route data to and from partner logical units in other domains.

When it runs as a VTAM application, WSim can simulate logical unit types 0, 1, 2, 3, 4, 6.1, 6.2, and 7. These logical unit types can participate in primary and secondary logical unit half-sessions. You can simulate multiple as well as parallel half-sessions. Refer to Chapter 12, “Summary of logical unit (LU) types,” on page 85 for a description of these logical unit types.

Note that this configuration does not attempt to present entirely realistic terminal simulation timings since WSim performs the simulation entirely within its software. Various system components, such as the host processor, operating system, and even VTAM, know that these are not real physical terminals. However, WSim presents a logically realistic simulation of local SNA primary and secondary logical units to the other half-session.

### Using the VTAMAPPL configuration

You can use the VTAMAPPL configuration to simulate secondary logical units to test application programs and network resources. For example, you can use VTAMAPPL to simulate LU2 terminals accessing real TSO, IMS/VS, CICS/VS, or Conversation Monitor System (CMS) applications. Or, you can simulate primary logical units to test application prototypes. For example, you can simulate application programs that real terminals can log on to.

Figure 4 shows WSim in the same host processor as the application programs. It is not required that WSim reside in the same host as the real applications that are being tested. WSim can access applications in another host by using a channel-to-channel adapter or an NCP. WSim uses the transport mechanisms of VTAM and the rest of the network so that the traffic can be sent from WSim to any other node in the network.

When you use the VTAMAPPL configuration, you can conduct stress, performance, regression, function, or capacity planning tests. This configuration is especially suited for application testing, where functional specifications and regression problems are major concerns and network communications or operating system performance is not. You can test the latter two items in a larger systems test, after the application changes are fully tested as units, by using WSim with additional line or subarea simulations.

### Testing application development

You can use WSim in at least two ways during application development:

- By simulating secondary logical units, you can test VTAM applications.
- By simulating primary logical units, you can use WSim to simulate an existing application or as a prototype of a new application.

***Testing VTAM Applications:*** Using this easy method, WSim runs as a VTAM application program and simulates terminals as secondary logical units. The real VTAM application program being tested thinks there are live terminals in session. This allows you to run simple, quick, and precise tests and repeat them as needed without human operators.

***Using WSim as an Application Prototype:*** WSim can simulate a primary SNA logical unit through the VTAM API. You can use WSim as an application prototype before coding begins. You can code panels, command languages, and decision-making into the scripts. WSim automatically performs all message handling.

You can derive the following benefits from using WSim as an application prototype before beginning application coding:

- Real terminal operators can log on to the prototype and assess the design.
- WSim-simulated terminals can also log on to the prototype. Scripts can be developed and saved for later testing with the real program code.
- The prototype is easily updated with new panels, logic, error messages, and other changes.
- The prototype does not need to use complex programming to send and receive messages.

## CPI-C transaction program simulation

Figure 5 on page 32 illustrates the logical configuration of a network in which you want to simulate CPI-C transaction programs (TPs) that are allocating or accepting conversations with other TPs. These TPs can be client programs (programs that allocate outbound conversations and do not accept inbound conversations) or server programs (programs that accept an inbound conversation), or both (programs that both allocate outbound conversations and accept an inbound conversation). You would use this logical configuration to test transaction programs that you want to run on subsystems such as IMS/VS, Customer Information Control System/Virtual Storage (CICS*/VS) or TSO. This VTAM subarea could be part of a large network with many other nodes.

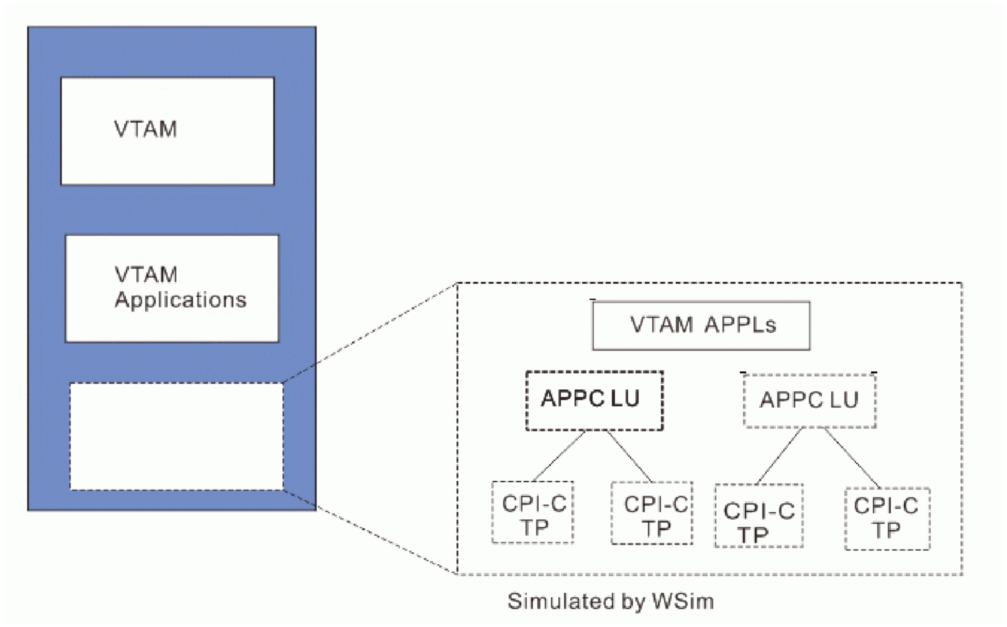


*Figure 5. Logical configuration for CPI-C transaction program simulation*

The physical configuration that corresponds to this logical configuration is the VTAM application (VTAMAPPL) configuration, which is shown in Figure 4 on page 30. In the CPI-C transaction program configuration, WSim runs as a VTAM application program and uses the VTAM APPC command API to simulate CPI-C LUs and TPs.

WSim does not drive any hardware directly when running exclusively as a CPI-C transaction program. Instead, it uses the VTAM API to send and receive messages, eliminating the need for any extra hardware resources. To run as a CPI-C transaction program without a communication controller, WSim requires only VTAM Version 3 Release 2, or later, and the VTAMLST APPL definitions in VTAM that enable WSim to communicate with VTAM.

WSim only simulates local CPI-C TPs within a VTAM subarea. However, this does not limit a simulated TP to same-domain conversations since VTAM may be able to route data to and from partner TPs in other domains.

Note that this configuration does not attempt to present entirely realistic TP simulation timings since WSim performs the simulation entirely within its software. Various system components, such as the host processor, operating system, and even VTAM, know that these are not real physical TPs. However, WSim presents a logically realistic simulation of local CPI-C TPs to other transaction programs.

### Using the CPI-C TP configuration

You can use the CPI-C TP configuration to simulate CPI-C client TPs to test CPI-C server TPs and network resources. For example, you can use the CPI-C TP support to simulate CPI-C client TPs that allocate conversations with real CPI-C server programs that run on IMS/VS or CICS/VS. Or, you can simulate server TPs to test client TP prototypes.

Figure 4 on page 30 shows WSim in the same host processor as the transaction programs. It is not required that WSim reside in the same host as the real TPs that are being tested. WSim can access TPs in another host by using a channel-to-channel adapter or an NCP. WSim uses the transport mechanisms of VTAM and the rest of the network so that the traffic can be sent from WSim to any other node in the network.

When you use the CPI-C TP configuration, you can conduct stress, performance, regression, function, or capacity planning tests. This configuration is especially suited for transaction program testing, where functional specifications and regression problems are major concerns and network communications or operating system performance is not. You can test the latter two items in a larger systems test, after the TP changes are fully tested as units.

### Testing application development

You can use CPI-C simulations in at least three ways during application development:

- By simulating client transaction programs, you can test real servers by providing them a conversation load.
- By simulating server transaction programs, you can test real clients by serving the conversation load the clients generate.
- By simulating both client and server transaction programs, you can integrate real clients or server TPs with simulated TPs. You can use this method also to simulate a load on the system or network for stress, performance, or capacity planning testing.

## TCP/IP application configuration

Figure 6 illustrates the logical configuration of a network in which you want to simulate TCP/IP clients in a TCP/IP network. The figure also illustrates the logical configuration of a network in which you want to simulate Simple TCP or UDP clients that are accessing an application through a server. You could use this logical configuration to do any of the following:

- Test Telnet/3270 and 3270E servers and 3270 applications by simulating Telnet/3270 and 3270E clients
- Test Telnet 5250 servers by simulating Telnet 5250 clients
- Test FTP servers by simulating FTP clients
- Test a variety of TCP applications and servers by simulating Simple TCP or UDP clients
- See the impact of increased TCP/IP traffic and numbers of clients on your TCP/IP network and servers.

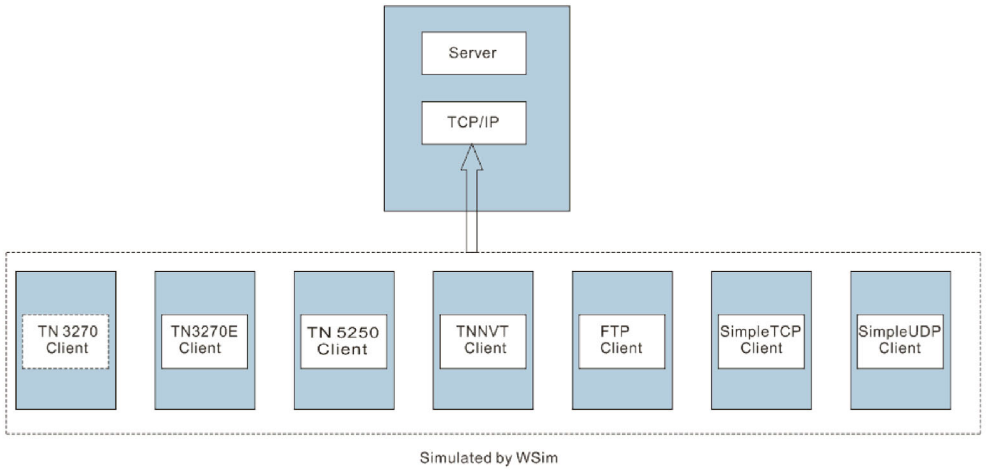


*Figure 6. Logical configuration for TCP/IP client simulation*

The physical configuration that corresponds to this logical configuration is the TCP/IP application configuration, shown in Figure 7.

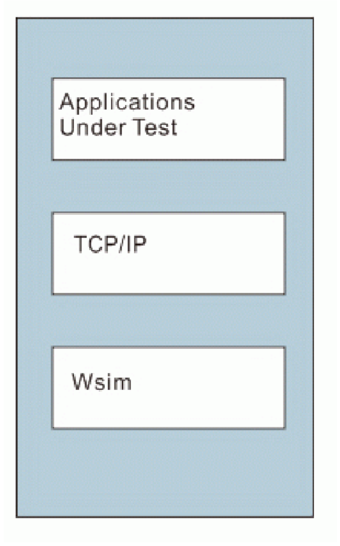


*Figure 7. Physical configuration for TCP/IP client simulation*

In the TCP/IP application configuration, WSim runs as a TCP/IP application program and uses the TCP/IP High Performance Native Sockets API to send traffic to your TCP/IP network.

WSim does not drive any hardware directly when running exclusively as a TCP/IP application. Instead, it uses the TCP/IP socket interface to the IBM TCP/IP for MVS product to send and receive messages, eliminating the need for any extra hardware resources. WSim requires the IBM TCP/IP for MVS product, Version 2 Release 2 or later.

## Using the TCP/IP application configuration

You can use the TCP/IP application configuration to simulate Telnet 3270, 3270E, 5250, NVT, FTP, or Simple TCP or UDP clients to test application programs and network resources. For example, you can use the TCP/IP application configuration to simulate Telnet 3270, 3270E, 5250, NVT, FTP, or Simple TCP or UDP clients accessing your mainframe applications through a TCP/IP network.

Figure 7 shows WSim in the same host processor as the application programs. This is not a requirement; WSim can access applications in another host. WSim uses the transport mechanisms of TCP/IP and the rest of the network so that the traffic can be sent from WSim to any other node in the TCP/IP network.

When you use the TCP/IP application configuration, you can conduct stress, performance, regression, function, or capacity planning tests. This configuration is especially suited for application testing, where functional specifications and regression problems are major concerns and network communications or operating system performance is not.

# Chapter 6. Defining the simulated network

This chapter describes what you need to know to define the network you want to simulate with WSim. You define the simulated network after you have configured your system.

WSim uses network definition statements and message generation decks to simulate the activities of a specific network. You use ***network definition statements*** to describe the devices in the network you want WSim to simulate. You use ***message generation decks*** to define the messages that are sent from the simulated resources in the network to the system under test. A ***script*** contains the network definition statements and one or more message generation decks. This chapter discusses how you use network definition statements to define the simulated network; Chapter 7, "Creating message generation decks," on page 45 describes the planning considerations for coding message generation decks. You can write STL programs to define the messages and use the STL Translator to convert these into message generation decks.

Defining the network and creating the message generation decks can be interdependent processes because the statements you code in the network definition can directly affect how and when messages are sent. For this reason, you may need to coordinate the coding of the network definition statements and the message generation decks. This is especially true if you are coding very complex scripts that contain a large network definition and many interrelated message generation decks. You can coordinate coding in the following ways:

- By modifying your network definition statements as you code the message generation decks
- By running a number of sample tests before you run the actual simulation. For more information about running sample tests, refer to "Running and analyzing a sample test" on page 61.

## Naming the network and its resources

Before you code any network definition statements, you should become familiar with all of the resources you want to simulate and how they are connected to each other and to the system under test. You need to assign names to the network itself and to the resources you want WSim to simulate. You code these names when you define the network and its resources. You use them when you need to refer to specific resources with other network definition statements or with operator commands. The names can contain 1 to 8 alphanumeric characters.

In general, you should assign a unique name to each resource in the network. This makes it easier to refer to that resource later. You must assign unique names to terminals to use some utilities, such as the Response Time Utility described in Chapter 9, "Using WSim output," on page 71. In addition, you should try to use a consistent and meaningful naming convention throughout the network. This helps you identify the location of the resource in the network. Finally, you may find it helpful to match the names in the simulated network to your real network.

## Using network definition statements

After you become familiar with what you want to simulate, you can code network definition statements to define the simulated network to WSim. Network definition statements specify the following information:

- Types of resources in the simulated network (for example, terminals or logical units)
- Characteristics of these resources (for example, if you are defining a terminal, you need to specify the terminal type)
- How these resources are connected to each other and to the system under test (for example, you specify which terminals are on which lines)
- Other features that control how the simulated network sends and receives messages, including the following:
  - Startup delays and message generation delays
  - Logic tests for the entire network that are in effect throughout the test
  - The order in which message generation decks are used
  - Whether you want message logging or message tracing.

The network definition statements you use depend on what logical configuration you are simulating; that is, you use different statements for VTAM application simulation, CPI-C transaction program simulation, and TCP/IP application simulation. This chapter briefly describes the types of statements you use for each of these types of simulation. For more information about coding the network definition statements for each of these types of simulation, refer to *Creating WSim Scripts*.

### Syntax of network definition statements

Network definition statements consist of the following three fields:

| | |
|---|---|
| **Name** | Contains a label to be used for the statement. |
| **Statement** | Contains the language statement. |
| **Operand** | Contains the operands for the statement. |

The following example shows the first statement in a network definition:

```
TESTNET NTWRK MSGTRACE=YES,OPTIONS=(CONRATE)
```

In this example, TESTNET in the Name field is the name you assign to the network. NTWRK in the Statement field indicates that you are specifying characteristics for an entire network. MSGTRACE=YES and OPTIONS=(CONRATE) in the Operand field specify that you want message tracing for the entire network and interval report message rates printed at the operator console when you run WSim.

For more information about the syntax of network definition statements, refer to the *WSim Script Guide and Reference*.

### Order of network definition statements

When you define the network you want WSim to simulate, you use different network definition statements depending on what part of the network you are describing and what type of network you are defining. You must arrange these statements in a particular sequence.

NTWRK is the first statement you use to define any network. This statement names the network and specifies characteristics that apply to the network as a whole. In addition, it specifies operands that establish defaults for lower-level statements in the definition. The other statements in the network definition follow the NTWRK statement in a prescribed sequence.

For example, if you want WSim to simulate SNA logical units that are accessing a VTAM application, you must arrange the statements in a particular order. The NTWRK statement comes first. Then, you use a VTAMAPPL statement to define a VTAM application program in the network. You use an LU statement after the VTAMAPPL statement to describe a logical unit half session. The final order of the network definition statements would look like this:

```
NTWRK
  VTAMAPPL
    LU
    LU
  VTAMAPPL
    LU
```

These statements also follow the NTWRK statement in a prescribed order. For information about these other statements, refer to the following section. For information about the correct sequence of network definition statements for each type of simulation, refer to *Creating WSim Scripts* and the *WSim Script Guide and Reference*.

## Simulating networks

The following section provides a brief overview of how WSim simulates different types of networks. It describes some of the network definition statements you use to define these networks and network resources to WSim. These networks and network resources include the following:

- Logical units (LUs) that access real VTAM applications using the VTAM Application Program Interface (VTAM application simulation)
- CPI-C transaction programs (TPs) that allocate or accept conversations with real TPs
- Telnet 3270 and 3270E clients that access real Telnet 3270 applications (Telnet simulation)
- Telnet 5250 and NVT clients
- File Transfer Protocol (FTP) clients that access real FTP servers (FTP simulation)
- Simple TCP and UDP clients that access various TCP servers and applications (Simple TCP Client simulation)

For complete information about each of these types of simulation, refer to *Creating WSim Scripts*. For complete information about the syntax of the network definition statements for each type of simulation, refer to the *WSim Script Guide and Reference*.

### Simulating logical units using the VTAM application program interface

This section discusses the network definition statements that define logical units that access real VTAM applications. These logical units could be terminals or other VTAM applications. For more information about how WSim supports VTAM application simulation, refer to "VTAM application simulation" on page 29 and to *Creating WSim Scripts*.

If you are simulating logical units accessing a VTAM application, you must use both of the following statements:

- VTAMAPPL statement
- LU statement.

### VTAMAPPL statement

Use the VTAMAPPL statement to define a VTAM application program. You must use either the Name field or the APPLID operand on the VTAMAPPL statement to specify the symbolic name of the application. This name must match an entry in VTAM's configuration tables (VTAMLST) that was created using an APPL definition statement in VTAM. The name specified is either the name of an active APPL resource or the value of the ACBNAME operand on an APPL statement.

You use the PASSWD operand on the VTAMAPPL statement to specify the password associated with the symbolic name of the VTAM application. The password you specify must match the value of the PRTCT operand on the APPL statement in VTAM.

In addition to the APPLID and PASSWD operands, you can specify other operands on the VTAMAPPL statement. These operands affect various aspects of the simulation, including message generation, message logging, SNA simulation, display simulation, and 3270 simulation.

### LU statement

Use the LU statement to define one or more logical unit half-sessions and to specify the type of half-session to be simulated. You can define primary or secondary logical unit half-sessions. In addition, you can specify a type of half-session for secondary half-sessions. Any number of LU statements can follow the VTAMAPPL statement. WSim simulates each logical unit half-session as a separate entity (such as a single display, device, or terminal) for message generation and message logging.

Other operands on the LU statement define logical unit half-session characteristics that are needed to interpret and generate unique data streams associated with products such as 3270 devices.

The example below shows a network definition for a VTAM application simulation containing a simulated secondary logical unit. The network is named VTAMNET, as defined on the NTWRK statement. INIT=SEC specifies that the simulated secondary logical unit initiates sessions with the system under test. The WSim logical unit is known to VTAM as WSIMLU, as defined on the VTAMAPPL statement. It is known to WSim as USER1, as defined on the LU statement. LUTYPE=LU2 specifies that the simulated logical unit is a Type two logical unit. RESOURCE=TSO specifies that the logical unit initiates a session with TSO in the system under test.

```
VTAMNET NTWRK INIT=SEC
WSIMLU  VTAMAPPL
USER1   LU    LUTYPE=LU2,
              RESOURCE=TSO
```

## Simulating CPI-C transaction programs

This section discusses the network definition statements that define simulated CPI-C transaction programs (TPs) that allocate or accept conversations with real TPs. Because WSim builds its CPI-C TP simulation on its VTAM application program interface, using a type 6.2 LU, the CPI-C network definition statements are similar to the logical unit definition statements. For more information about how WSim supports CPI-C TP simulation, refer to "CPI-C transaction program simulation" on page 31 and to *Creating WSim Scripts*. For more information about how WSim simulates logical units, see "Simulating logical units using the VTAM application program interface" on page 39.

If you are simulating CPI-C transaction programs, you must use both of the following statements:
- APPCLU statement
- TP statement.

### APPCLU statement

Use the APPCLU statement to define a CPI-C logical unit. You must use either the Name field or the APPLID operand on the APPCLU statement to specify the symbolic name of the logical unit. This name must match an entry in VTAM's configuration tables (VTAMLST) that was created using an APPL definition statement in VTAM, and the matching entry in VTAMLST must specify APPC=YES. The name specified is either the name of an active APPL resource or the value of the ACBNAME operand on an APPL statement.

You use the PASSWD operand on the APPCLU statement to specify the password associated with the symbolic name of the VTAM application. The password you specify must match the value of the PRTCT operand on the APPL statement in VTAM.

In addition to the APPLID and PASSWD operands, you can specify other operands on the APPCLU statement. These operands affect various aspects of the simulation, including message generation, message logging, the number of sessions between the TPs that can be active concurrently, and CPI-C side information that you want to apply to all TPs that are defined for this LU.

If your network definition also includes VTAMAPPL statements (to define logical units that access VTAM applications), all APPCLU statements must precede the VTAMAPPL statements.

### TP statement

Use the TP statement to define one or more transaction programs and to specify whether the transaction program is to be a client or a server TP. Any number of TP statements can follow the APPCLU statement. WSim simulates each transaction program as a separate entity for message generation and message logging.

The example below shows a WSim network definition for a CPI-C TP simulation containing a simulated client TP. The network is named CPICNET, as defined on the NTWRK statement. The CPI-C logical unit is known to VTAM as LU1, as defined on the APPCLU statement. The transaction program is known to WSim as TP1, as defined on the TP statement. TPTYPE=CLIENT specifies that the simulated transaction program is a client transaction program.

```
CPICNET NTWRK
LU1     APPCLU
TP1     TP    TPTYPE=CLIENT
```

## Simulating TCP/IP clients

WSim can establish TCP/IP connections that allow simulated clients to access servers and applications through a TCP/IP network. For this type of simulation, WSim runs as one or more Telnet 3270, 3270E, 5250, NVT, FTP, or Simple TCP or

UDP clients on the driving system that connect to one or more servers on foreign (or local) hosts. For more information about TCP/IP client simulation, refer to *Creating WSim Scripts*.

To define TCP/IP clients to be simulated, use the following statements:
- TCPIP statement
- DEV statement.

### TCPIP statement

Use the TCPIP network definition statement to define a connection to the IBM TCP/IP for MVS product on the local host. This connection allows WSim to use the sockets interface to communicate data over a TCP/IP network. The TCPNAME operand specifies the TCP/IP virtual machine or address space name on the local host. You can also specify other operands to provide defaults to devices defined on this connection.

### DEV statement

The DEV statement defines a TCP/IP client to be simulated. The TYPE operand is used to specify which type of client is to be simulated - TN3270 (Telnet 3270), TN3270E (Telnet 3270E terminal), TN3270P (Telnet 3270E printer), TN5250 (Telnet 5250), TNNVT (Telnet NVT), FTP (File Transfer Protocol), STCP (Simple TCP), or SUDP (Simple UDP). Multiple clients of varying types may be defined for a single TCPIP statement. Multiple TCPIP statements with one or more DEV statements each may also be used. Each client can connect to a different server that can be specified using the SERVADDR operand. The PORT operand specifies which TCP/IP port is to be used when establishing the connection for this device. For Telnet 3270, 3270E, 5250, NVT, and FTP, the default PORT is the well known port for the protocol being simulated. For Simple TCP and UDP, the default PORT is 256. The FTP protocol normally requires use of the port specified or defaulted and the next lower port number as well.

The example below shows a network definition for a TCP/IP network that contains a 3270 device that will log on to the foreign host designated by IP address 9.67.25.1, and will use port 1021, an FTP client that will access an FTP server on a foreign host designated by IP address 9.67.44.1, and a Simple TCP client that will connect to a server designated by IP address 9.67.43.62, using port 5555.

```
TCPIP1 NTWRK
CONN1  TCPIP
DEV1   DEV   TYPE=TN3270,
             SERVADDR=9.67.25.1,
             PORT=1021
DEV2   DEV   TYPE=FTP,
             SERVADDR=9.67.44.1,
DEV3   DEV   TYPE=STCP,
             PORT=5555,
             SERVADDR=9.67.43.62
```

## Simulating specific devices

In addition to defining different types of networks, network definition statements can define specific types of devices. As explained in *Creating WSim Scripts*, when you use network definition statements to define the following devices, you may need to consider special coding requirements:
- IBM 3270 Information Display System
- IBM 5250 Display Station.

These coding requirements can include special considerations for the following:

- Conditions for message generation
- Device-specific special features.

Refer to *Creating WSim Scripts* for information about the special requirements for specific devices.

## Coding network options

You use optional statements and operands in the network definition to control many aspects of the simulation. These include the time frame used to start devices in the network, the order in which message generation decks are used, and the method used for logging the messages sent and received by WSim during the simulation. For information about coding these network options, refer to *Creating WSim Scripts* and the *WSim Script Guide and Reference*.

# Chapter 7. Creating message generation decks

Message generation is the process by which simulated terminals send and receive messages. You use message generation decks to control the messages sent and actions taken when messages are received by a simulated terminal. This chapter discusses what you need to consider before you create message generation decks, describes the methods you can use to create them, and suggests procedures for testing scripts.

For complete details about how to create message generation decks using these methods, refer to the following books:

- *Creating WSim Scripts*
- *WSim Script Guide and Reference*
- *WSim Utilities Guide*

**Note:** Unless otherwise stated, the discussions in this chapter apply to all simulated terminal types that can generate messages. The terminal type that cannot generate messages is LU3. The books in the preceding list also give device-specific information about message generation.

## What are message generation decks?

***Message generation decks*** define the messages that WSim generates for a simulated terminal. A message generation deck is a collection of statements that enables you to simulate the actions of a terminal and a terminal operator. You can use message generation decks to generate messages, set intermessage delays, define logic tests, define and control event actions, and perform miscellaneous functions such as saving data for later use.

You can code as many or as few statements as you want within a single message generation deck. In addition, a given terminal can use any number of different message generation decks in any order you want. For this reason, you can use message generation decks to group test data for a terminal into logical transactions or other units of work.

## Steps for creating message generation decks

You use the following general steps to create message generation decks:

1. Decide which transactions to test.
2. Decide which application files and what data to use.
3. Create the message generation decks using one of the available methods and combine them with the network definition to form a script.
4. Test the complete script and modify or revise as needed.

The following sections describe each of these steps in more detail.

## Deciding which transactions to test

Before you create the message generation decks for your WSim test, you need to decide which transactions to test. In WSim, a ***transaction*** is an exchange of data between a simulated resource and the system under test. This data exchange accomplishes a particular action or result. For example, a transaction might involve recording a customer's deposit and updating the account balance.

The transactions you test depend on the objectives of your test. If you are testing hardware connections or system software, you are usually not concerned about the type of transactions you use. Any transaction between the simulated resources and the system under test is probably adequate for the test. In contrast, if you are testing an application, you need to select the transactions you want to test carefully. You usually do not need to test all of the transactions in the application; however, you should try to test the most important ones. These could be those transactions that:

- Take the most host processor time (for example, sorting).
- Generate the most messages. Remember the 80-20 rule: 80% of the messages probably come from 20% of the transactions.
- Are the most important for your organization. For example, if you are testing a banking application, you might want to test transactions involving the automatic teller machines.

In addition to deciding which transactions you want to test, you need to consider the following:

- The content of the messages you want to send
- The messages you expect to receive back
- The transaction mix (that is, the order in which the transactions are executed and which terminals participate in each transaction)
- The transaction rate.

The following sections briefly describe each of these considerations.

### Considering what messages you want to send

After you determine the transactions you want to test, you need to consider what kinds of messages you want the simulated resources to send to the system under test. The content of these messages depends on the following:

- The types of transactions you are testing
- The types of terminals WSim is simulating.

For example, suppose you want WSim to simulate a 3270 terminal accessing a banking application. In this example, a typical transaction might include entering a deposit and updating the account balance. The actual data sent to the banking application is the account number, the amount of the deposit, and so on. However, the message generation deck for this transaction would also need to indicate such user actions as pressing the Tab key to move to the next field or pressing the Enter key to send a completed panel.

In other words, before you can code the message generation decks, you need to determine what the user would type in order to complete the transaction. This information could be actual data (for example, account numbers or text strings such as Hello. How are you?) or it could represent the type of information that is sent to the system under test when a user presses a specific key on the keyboard (for example, Tab or Enter).

## Considering what messages you expect to receive

You also need to decide what messages you expect to receive back from the system under test. You can test for the expected responses by coding logic tests into your scripts. These logic tests consist of statements that test for a certain response from the system under test and take a particular action based on the response. For more information about logic tests, refer to *Creating WSim Scripts* or *WSim Script Guide and Reference*.

**Note:** Logic tests can also test the messages sent by a simulated resource to the system under test.

## Considering the transaction mix

In addition to considering the messages you want to send and expect to receive, you also need to consider the transaction mix. The transaction mix is the order in which WSim executes the message generation decks as well as which terminals use which decks. For example, if you are conducting a stress test, you may want some terminals to log on to and off of the operating system repeatedly, while other terminals are completing more complex transactions. You can specify this mix of transactions in the network definition by using PATH statements and the PATH operand. The PATH statement specifies the order in which the decks will be executed, while the PATH operand indicates which paths a specified resource will execute.

In the example below, the path named SIMPLE specifies that the INITSESS deck will be processed before the LOGOFF deck. The path named COMPLEX specifies that the INITSESS deck will be processed before the message generation decks named ALLOC, EDIT, DELETE, and LOGOFF. The PATH operands on the DEV statements indicate that DEVICE1 will execute the SIMPLE path and DEVICE2 will execute the COMPLEX path. Note that the vertical ellipses indicate that there may be other network definition statements that are not shown in this example.

```
TESTNET1 NTWRK
          .
          .
          .
SIMPLE   PATH  INITSESS,LOGOFF
COMPLEX  PATH  INITSESS,ALLOC,EDIT,DELETE,LOGOFF
          .
          .
          .
DEVICE1  DEV   PATH=(SIMPLE)
DEVICE2  DEV   PATH=(COMPLEX)
```

The paths that you define are selected and executed repeatedly. When a terminal has executed the last deck in the last path defined for it, it starts again with the first deck in the first path. Each terminal maintains its own position in the path and is not affected by other terminals. However, you can use BRANCH, CALL, and IF statements in your message generation decks to alter this sequence. You can also define the order in which WSim selects the message generation decks listed on a PATH statement. You can define a certain order, a random order, or a probability distribution.

When you conduct a test, start by coding a simple transaction. This transaction may include one or more message generation decks. Combine this transaction with a simple network definition and run a sample test to ensure that the transaction works correctly. Then, add additional transactions and more complex network definitions until you have the transaction mix you want. Be sure to verify each transaction separately before combining it with other transactions.

For more information about using the PATH statement and the PATH operand, refer to *WSim Script Guide and Reference* and *Creating WSim Scripts*.

### Considering the transaction rate

Finally, you need to consider the rate of transactions. With WSim, you can create increasingly complex situations by coding intermessage delays. An ***intermessage delay*** is the period of time that WSim waits, or delays, between the messages it sends to the system under test. You can use these delays to simulate the delays that real operators make as they view the screen, think about the information, and enter more data.

WSim provides a number of network definition and message generation statements for defining intermessage delays. You can control intermessage delays for a specific resource or for the entire network. You can even define delays on a message-by-message basis. In addition, you can specify multiple intermessage delays with the UTI statement. For a complete explanation of how to code intermessage delays, refer to *Creating WSim Scripts* or *WSim Script Guide and Reference*.

## Deciding which application files and data to use

After you decide which transactions to test, you need to decide what application files you will use and what data should go into those files. You may want to use a copy of your production database for the test; however, you can often conduct the test using a small test database that contains a representative sample of the production database. The data that you include in this test database depends on the objectives of your test. For example, if you are testing a full-screen application, you may want to provide the application with a file that it can edit during the test.

## Creating message generation decks

After you have decided what transactions you want to test and what application files and data you want to use, you are ready to create the message generation decks for your test. First, spend some time listing the steps in the transactions you are testing. You can list these steps using ordinary English statements or pseudocode. Listing the steps helps ensure that you completely understand the transactions before you spend time coding message generation decks. It also provides you with a detailed set of instructions that real terminal operators can follow if you decide to trace system activity.

Then, decide what method to use to create the message generation decks. You can use any of these methods to create message generation decks for WSim:

- Write message generation statements
- Write Structured Translator Language (STL) programs and use the STL Translator to generate message generation decks
- Use one of the script generating utilities available with WSim to convert captured data traces into message generation decks or create actual messages. WSim script generating utilities include:

  - Interactive Data Capture (also produces STL programs)
  - Script Generator Utility
  - ITPLU2RF
  - 111 TCP/IP Trace STL Generation Utility

The method you use depends on what you are testing as well as the following:
- How familiar you are with WSim. If you are a new user, you should use STL and the STL Translator or one of the script generating utilities.
- What kind of messages you want WSim to send to the system under test. For example, if you want to simulate a number of real users using your application, you might trace actual system activity and use the Script Generator Utility to convert the trace records into message generation decks.
- What stage of development your application is in.

Note that you may need to use several of these methods to create message generation decks for your simulation. For example, the message generation decks created by the Script Generator Utility generally need to be modified before you can use them. You can use message generation statements to do this.

The following sections describe each of the methods in more detail and provide information to help you select the best method for your situation. For more information about using each of these methods, refer to *Creating WSim Scripts*, *WSim Script Guide and Reference*, or *WSim Utilities Guide*.

## Using message generation statements

As explained in the beginning of this chapter, message generation decks are composed of message generation statements. In most situations, you can use STL or one of the script generating utilities to create the message generation decks for your test, so you do not need to code the actual message generation statements. However, you need to understand these statements to interpret the output from the STL Translator and the script generating utilities. This is especially true if you are trying to debug your scripts.

In addition, there are certain instances when you need to code message generation statements by hand. These instances include the following:
- When you are modifying the output created by one of the script generating utilities
- When you want to incorporate additional message generation decks in a script produced by the STL Translator or one of the script generating utilities
- When you want to send certain types of messages from a non-3270 device
- When you want to set up SNA chaining explicitly
- When you want to modify message generation decks created with a previous version of WSim.

The syntax for message generation statements is the same as for network definition statements. Refer to “Using network definition statements” on page 38 or the *WSim Script Guide and Reference* for an explanation of the syntax of message generation statements and network definition statements. For more information about coding message generation statements, refer to *Creating WSim Scripts*.In addition, *Creating WSim Scripts* includes several examples that may help you understand how to code these statements.

After you have coded your message generation statements, you should preprocess them using the Preprocessor. The Preprocessor checks the syntax of the statements and stores them in data sets for use during the simulation. For more information about the Preprocessor, see “Preprocessing the script” on page 61.

## Using the Structured Translator Language and the STL Translator

The Structured Translator Language (STL) is a high-level, structured programming language that you can use to create message generation decks and define terminals and devices you want to simulate using WSim. Like other structured languages, STL uses constants, variables, expressions, and structured control statements in its programs.

Whether you are a new or experienced user, you may find it convenient to write new message generation decks using STL. STL enables you to program using familiar programming structures and conventions. Because of its similarity to other programming languages, STL is easy to learn and it enables you to write message generation decks quickly and accurately. If you like, you can incorporate message generation statements into STL programs.

Typically, an STL program is divided into one or more procedures, which are similar to subroutines in other programming languages. The STL Translator translates STL programs into message generation decks. Each message generation deck is equivalent to one STL procedure. Thus, an STL program can be translated into more than one message generation deck.

**Note:** You can also include network definition in STL input data sets. The STL Translator invokes the Preprocessor to validate and store these statements for you.

### Sample STL program

Below shows two sample STL programs each containing one STL procedure. Each procedure begins with an MSGTXT statement and ends with an ENDTXT statement. The name of the procedure appears in front of the MSGTXT statement.

```
/* STL program used to log terminal on to MYAPPL. */
Logon: Msgtxt
Initself('MYAPPL')
Endtxt

/* STL program used to test message generation. */
Test1: Msgtxt
Do i = 1 to 5
   Type "Hello"
   Transmit using PF1,
      and Wait until onin substr(screen,4,17) = "Hello to you too."
End
Endtxt
```

In this example, the first STL procedure, named LOGON, defines the text that a terminal would use to log on to an application named MYAPPL. The second STL procedure, named TEST1, defines messages that you might use to test message generation. When it executes TEST1, WSim simulates a user typing “Hello” and then pressing PF1 to send the message to the application. WSim would then wait until the application's response “Hello to you too.” appeared at position 40 on the screen. The DO statement indicates that these messages should be sent and received five times.

Before you can use this STL program with WSim, you must use the STL Translator to translate it into message generation statements. You then combine these message generation decks with network definition statements to form a script for the simulation. For more information about writing STL programs, refer to *WSim Script Guide and Reference*.

### Using the STL Translator

The STL Translator is a utility that translates STL programs into message generation decks. The input to the STL Translator consists of an STL source data set. You can run the STL Translator using the WSim/ISPF Interface, JCL, or a TSO CLIST. *WSim Script Guide and Reference* provides examples of running the STL Translator in each of these situations and discusses the execution parameters you use to run the utility.

The output from the STL Translator consists of the following:

- Printed listing of the message generation statements and the STL source statements.
  1. STL Translator listings:
     - Includes all error messages
     - Includes source statements, unless NOSOURCE is specified
     - Includes message generation statements, unless NOWSIM is specified
     - Includes a variable dictionary
     - Includes a statistics report
  2. Preprocessor listings:
     - Includes all error messages
     - Includes the Preprocessor listing, unless NOLIST is specified
     - Includes a cross-reference report, unless NOXREF is specified
     - Includes a network definition summary, if SUMMARY is specified.
- Optionally, INITDD partitioned data set members containing network definition statements.
- Optionally, MSGDD partitioned data set members containing message generation statements. These data set members do not need to be preprocessed by the Preprocessor before you use them to run WSim.
- Optionally, a sequential output file containing the message generation statements and the STL source statements.
- Optionally, a partitioned data set member containing the STL variable table and a listing of the correlations between the STL statements and the message generation statements. The contents of this data set member are used by the Loglist Utility when it prints STL trace records and by the Q operator command to display the STL statement numbers. (Refer to “Using STL trace records” on page 57 for information about STL trace records.)

*WSim Script Guide and Reference* provides more information about using the output produced by the STL Translator.

## Using the Interactive Data Capture Utility (ITPIDC)

The Interactive Data Capture Utility (IDC) ITPIDC is a host application that gives you an easy way to interactively capture 3270 device session data and generate scripts. To use Interactive Data Capture, you simply log on to it, much like any other VTAM application. You can also start IDC on your TSO ID and utilize the TSO console. Then, through Interactive Data Capture, log on to the VTAM application you want to test and perform the actions you want WSim to simulate. Interactive Data Capture remains transparent to your application while it captures session traffic.

When you are done, you tell Interactive Data Capture to generate either WSim message generation decks or an STL program from the captured data.

## Using the Script Generator Utility

The Script Generator Utility enables you to create message generation decks that are based on traces of real users using a real application. To use this utility, you must put the captured trace in a specified format and sort it by resource name, date, and time. You then use the sorted trace as input to ITPSGEN, the utility program that actually generates the message generation decks.

When you use the Script Generator Utility, you follow these five steps:

1. Obtain a trace of users using your application.
2. Reformat the trace output if it is not in the format required by the Script Generator Utility.
3. Sort the reformatted output using any standard sort program.
4. Define the network using network definition statements.
5. Generate the message generation decks for that network using ITPSGEN.

Figure 8 on page 53 shows the programs you can use for each of these steps. The following sections describe these steps in more detail. Refer to *WSim Utilities Guide* for complete information about these steps.

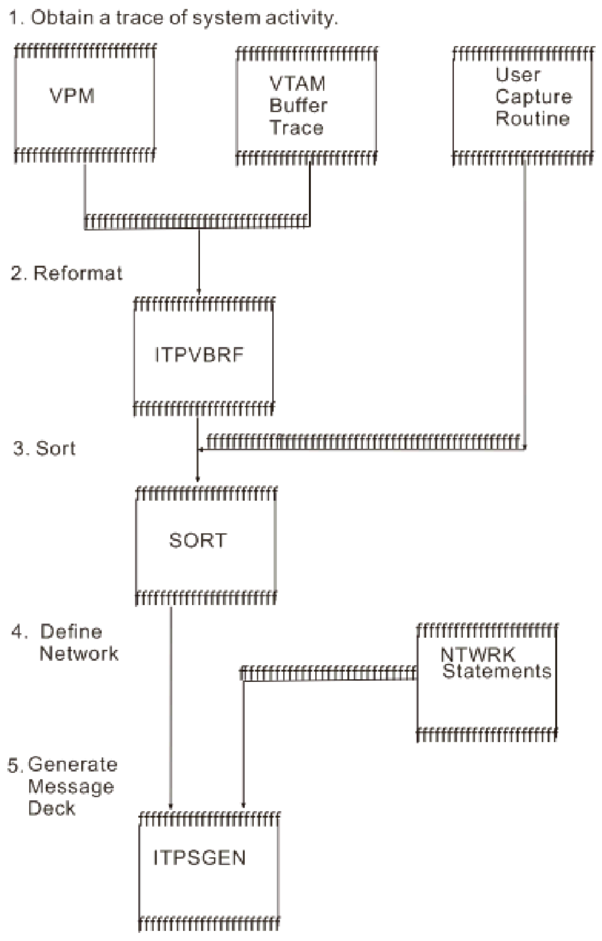


*Figure 8. Generating scripts with the script generator utility*

## Obtaining a trace of system activity

As indicated in Figure 8, you can use several methods to obtain a trace of system activity:

- The NetView® Performance Monitor (NPM) captures path information units (PIUs) for selected logical units.
- The VTAM Buffer Trace with the Generalized Trace Facility (GTF) traces system activity.
- User-written capture routines trace other forms of system activity.

## Reformatting the trace output

If the trace of system activity is not in the format required by ITPSGEN, you must reformat it before you can use it to create message generation decks. WSim provides a program (ITPVTBRF) to help you reformat trace output. For more information about this program, refer to *WSim Utilities Guide*.

## Sorting the trace data

Before you use ITPSGEN, you must sort the trace data into ascending order based on the name, date, and time fields. You can sort the data using any standard sort program.

## Defining the network

The network definitions you use as input to ITPSGEN should be complete, syntactically correct networks. For more information about checking the syntax of statements, refer to “Preprocessing the script” on page 61.

ITPSGEN uses the network definition statements to determine the terminal names for which decks are generated. These names, taken from the DEV and LU statements in the network definitions, must correspond to the resource names used in the trace data set. For this reason, when you use ITPSGEN, be sure that your network definition statements do not contain duplicate names.

## Generating message generation decks with ITPSGEN

ITPSGEN creates message generation decks from sorted trace data sets. It can also update the network definitions to reflect the message generation decks generated. To run ITPSGEN, you need a sorted trace data set and at least one WSim network definition. You can also include control statements and additional network definition statements. For examples of the WSim/ISPF Interface, JCL and TSO CLISTs needed to run ITPSGEN, refer to *WSim Utilities Guide*.

The output from ITPSGEN includes the message generation decks as well as reports and updated network definitions. In addition, ITPSGEN can create a sequential data set that contains all message generation decks and network definitions.

## Capturing terminal traffic

The quality of the scripts created by the Script Generator Utility cannot be better than the actual traffic traced. In other words, if you trace system activity that does not represent the type of use you want to test, the scripts created by the Script Generator Utility may not be useful. For example, you may not want to trace system activity early in the day (when many users are logging on to the system), at lunch time (when many terminals are unattended), or at the end of the day (when many users are logging off of the system).

The following section describes two methods for capturing the messages sent from a terminal. For more information about these methods, refer to *WSim Utilities Guide*. That book also gives solutions for typical problems you may encounter when you use the Script Generator Utility.

**Note:** The Script Generator Utility creates a script that is based on a specific functional level of the application. For this reason, changing the application may make the script unusable.

***Capturing Single Terminal Traffic:*** This method enables you to test a particular set of transactions and is easily controlled. You would use this method if you wanted to test specific application functions.

To capture traffic from a single terminal, follow these steps:

1. Identify a single terminal and operator whose traffic you want to trace.
2. Start the capture routine.
3. Have the operator log the terminal on to the application, execute the transactions, and log off.
4. Stop the capture routine.
5. Reformat and sort the captured data.
6. Define the network using network definition statements.

7. Run ITPSGEN to obtain one message generation deck that contains all of the messages sent by that terminal.
8. Modify the network definition statements and message generation deck, if desired, and create a script.
9. Run WSim using your script as input.

You may want to modify the message generation deck produced by ITPSGEN to ensure that the simulated terminals do not access the same data base records. To do this, first divide the deck into smaller decks that contain only one transaction each. This makes it easier to create transaction mixes. Then, add some variable data to the decks. You can do this by finding where a user variable (such as an account code or part number) was entered and replacing it with a reference to a user table. You can select entries from a user table sequentially, randomly, or by using a probability distribution. For more information about using user tables, refer to *Creating WSim Scripts* and *WSim Script Guide and Reference*.

***Capturing System Terminal Traffic:*** This method enables you to capture actual traffic sent by the all the terminals in the system. You might use this method if you wanted to conduct a stress test on your application.

To capture all the terminal traffic within the system, follow these steps:
1. Start the capture routine.
2. Start the host application.
3. Capture the terminal traffic for a specified amount of time.
4. Stop the capture routine.
5. Reformat and sort the captured data.
6. Define the network using network definition statements.
7. Run ITPSGEN to produce the message generation decks for each terminal.
8. Modify the network definition statements and message generation deck, if desired, and create a script.
9. Run WSim using your script as input.

Although capturing system terminal traffic can be a very useful method, there are several potential disadvantages to using it. First, tracing many terminals at once may adversely affect the performance of your system. Second, the transactions entered by the terminal operators may not be the type required for the test. This is because the actions of each terminal are not under your control. Finally, the scripts generated from these traces may not work correctly unless you modify them. This is because the traces capture the data that is received by the host. WSim, on the other hand, defines keystrokes. Since ITPSGEN tries to re-create keystrokes based on the data sent to the host, keystrokes that are not sent to the host (such as a local clear) will not be accounted for in the message generation decks. As a result, the script may not work as planned.

## Using SNA 3270 Reformatter Utility (ITPLU2RF)

ITPLU2RF is a batch utility that reformats NPM log records (FNMVLOG) from LU type 2 sessions into log records. You can then run the following utilities directly:

| | |
|---|---|
| **ITPLSGEN** | Create STL programs or message decks. |
| **ITPLL** | Create a loglist. |
| **ITPCOMP** | Compare log display records. |
| **ITPRESP** | Create response-time reports. |

ITPLU2RF also provides reports about the LU type 2 sessions from the NPM log, showing statistics on records counts, data lengths, and time stamps.

## 111 Using the TCP/IP Trace STL Generation Utility

111 You can create STL programs from a TCP/IP trace by using the TCP/IP Trace STL
222 Generation Utility. The TCP/IP trace contains records for HTTP messages that are
111 exchanged between a client and a server that runs on z/OS.

111 To help you obtain a TCP/IP data trace of the communication between a client and
111 server, WSim provides the TCP/IP Data Trace Utility (ITPIPTRX).

### 111 Obtaining a TCP/IP trace

111 You can start and stop TCP/IP data and packet traces by using the **V TCPIP**
111 command that is issued from a z/OS system console.

111 If you cannot issue the **V TCPIP** command, you can use the TCP/IP Trace Utility
111 (ITPIPTRX) to create a TCP/IP data trace of the communication between a client
111 and server.

111 For more information on TCP/IP Trace Utility, refer to the *WSim Utilities Guide*.

### 111 Generating STL from a TCP/IP trace

111 The TCP/IP Trace STL Generation Utility program ITPIPGEN creates an STL
222 program from the trace records for the HTTP messages that are exchanged
111 between a client and a server that is running on z/OS.

111 For further information on running ITPIPGEN, refer to *WSim Utilities Guide*.

## Testing scripts

Once you have created the message generation decks for your simulation, you need to test them to ensure that they are coded correctly and that they function as you intended. You can check statement syntax by using the Preprocessor or the STL Translator. (Refer to “Preprocessing the script” on page 61 and to *WSim Utilities Guide* for information about the Preprocessor; refer to *WSim Script Guide and Reference* for information about the STL Translator.) Then, you can ensure that the message generation decks function as you intended by using one or more of the following methods during the simulation:

- Message trace records to trace the message generation process
- STL trace records to trace the message generation process for STL programs
- Self-checking scripts.

The following sections briefly describe each of these methods.

## Using message trace records

You can use the message trace (MTRC) records in the log data set to trace the activity caused by specific message generation statements. To obtain these records, you must take the following steps:

1. Specify MSGTRACE=YES in the network definition or with the A (Alter) operator command. This causes message trace records to be written to the log data set.

2. Use the Loglist Utility to format and print these records.

Message trace records show the steps taken through the message generation deck. They also indicate the action or inaction of logic test IF statements. You can use these records when you are trying to test and debug message generation decks and the associated logic tests. They are especially useful when you are learning to write scripts. For more information about using these records, refer to *Creating WSim Scripts*. For more information about using the Loglist Utility, refer to *WSim Utilities Guide*.

## Using STL trace records

You can use the STL trace (STRC) records in the log data set to trace the activity caused by specific STL statements. To obtain these records, you must take the following steps:

1. Request that the STL Translator create correlation records by using the PROGRAM= execution parameter or the @PROGRAM statement in your STL program. These records correlate a message generation statement with the STL statement that produced it.
2. Specify STLTRACE=YES in the network definition or with the A (Alter) operator command. This causes STL trace records to be written to the log data set.
3. Use the Loglist Utility to format and print the records.

STL trace records are similar to message trace records and can be used when you are trying to test and debug message generation decks that were created with STL and the STL Translator. These records are especially helpful when you are learning to write STL programs. For more information about STL trace records, refer to *WSim Script Guide and Reference*. For more information about using the Loglist Utility, refer to *WSim Utilities Guide*.

## Using self-checking scripts

Although you should test and debug your scripts as you write them, doing so is not always enough to ensure that unexpected situations will be handled correctly. Consider, for example, some of the problems that can occur during a test:

- The logon can fail. This could be caused by new or expired passwords, too many users, unavailable applications, unavailable routes, and so on.
- A message from the operator can appear on a simulated VM panel. This causes the screen to enter the HOLDING state, so subsequent transactions are not accepted.
- The system can change between two test runs. This could be caused by a problem with a data set, a different response message, or authorization modifications.
- A timing difference can exist because of a heavily loaded host processor. If the system does not respond fast enough, another message can be transmitted by the terminal causing the session to become out of synchronization.

These are just a few simple examples of problems that can arise during the simulation. Self-checking scripts help guard against unexpected problems by verifying that a terminal session is proceeding as expected.

You do not need to use self-checking scripts for all simulations. However, you should consider the consequences of not using them. In other words, what happens if you do not notice a simulation problem in a timely manner? If you are running short and simple simulations, you can probably detect errors yourself very easily. On the other hand, if an eight-hour test run is wasted because terminals are out of synchronization, then preparing self-checking scripts may be well worth the effort.

In these situations, you should use self-checking scripts to guard against wasted time. In addition, you should use self-checking scripts when you are testing the functions of new or changed applications. Self-checking scripts can assist you in debugging these applications by doing the following:
- Detecting errors sooner than otherwise possible
- Detecting errors that otherwise would not be detected.

## Writing self-checking scripts

When you create a self-checking script, you add IF statement logic tests to the script to check for the expected response and to take action if an unexpected response is received. The action can be as simple as stopping the device or as complex as choosing from several possible courses of action based on what was actually received. You can code these logic tests in hand-written message generation decks as well as in decks created by STL or one of the script generating utilities. In addition, you can code logic tests in the network definition.

Below shows an example of a message generation deck with self-checking logic that causes the system to wait for the expected response. In this example, WSim sends Message 1 to the system under test and waits until Reply 1 is received before continuing. After receiving Reply 1, Message 2 is generated and sent to the system under test. WSim waits for Reply 2 before continuing to the end of the message deck.

```
DECK5 MSGTXT
*                          Beginning of DECK5.
MSG1  TEXT  (MESSAGE 1)    Defines MSG1.
0     IF    LOC=B+0,
      TEXT=(REPLY 1),
      THEN=CONT
WAIT1 WAIT
MSG2  TEXT (MESSAGE 2)     Defines MSG2.
0     IF   LOC=B+0,        Defines basic logic test.
           TEXT=(REPLY 2),
           THEN=CONT
WAIT2 WAIT
      ENDTXT               End of DECK5.
```

For a complete explanation of how to produce self-checking scripts by coding logic tests in the message generation decks, refer to *Creating WSim Scripts* or *WSim Script Guide and Reference*. *Creating WSim Scripts* provides more information about coding IF statements that apply to the entire network.

## Detecting problems without self-checking scripts

There are a number of indicators you can use to track the progress of a test without using self-checking scripts. However, as explained in the following list, each of these methods has potential disadvantages:
- The Loglist Utility. This utility tells exactly how the simulation ran. Its primary drawback is that you run it after the simulation is over. In addition, the output from the Loglist Utility is sometimes too long to read in complete detail. Therefore, problems can go undetected.
- The Log Compare Utility. This utility enables you to compare 3270 display records from two log data sets. You can use it to detect incorrect data, data in the wrong position, and panels in the wrong order. Its primary drawback is that

you run it after the simulation is over. In addition, you must have log data sets from two different WSim runs to detect any problems.

- Display Monitor Facility. This facility enables you to see the display image of a simulated 3270 display device or logical unit Type 2 terminal during the simulation. You can use it to monitor simulations involving those terminal types; however, it may be impractical as a verification tool if large numbers of terminals are being simulated.
- Message rates. WSim can display send-and-receive message rates during the simulation. If these rates are high and remain stable, then everything may be running as expected. However, message rates cannot tell you if the simulated terminals are out of synchronization.
- Query command output data. You can use the Q operator command to query a terminal during the simulation; however, it is unlikely that you will query your terminal at the exact instance that a problem occurs.

  You can also use the G (Terminal Status Query) operator command during the simulation to find out which terminals are active, inactive, quiesced, terminated, or ready. However, the G command cannot tell you why a terminal is inactive or how long it has been inactive.
- Terminal hang situations. If you notice that a terminal has unexpectedly stopped sending and receiving messages, you know that there is a problem. However, it may take appreciable effort to find the cause of the problem and correct it.
- Operating system messages. Messages on the operating system console can warn you about some problems, but they cannot tell you what the problem is.
- Scan function. When you code the SCAN operand on the NTWRK statement, WSim examines all of the terminals and devices in the network each minute. WSim can notify the operator of inactive terminals. Although you can code message generation decks that will attempt to recover inactive terminals automatically, using the scan function does not tell you why the terminal became inactive.
- Application program abnormal ending. This is a sure sign of a problem. However, more subtle problems can exist without causing abnormal endings.

# Chapter 8. Running the test

This chapter explains the sequence to follow when you run WSim. It discusses the following tasks:

- Running and analyzing a sample test
- Preprocessing the script
- Estimating storage requirements
- Running WSim on MVS and TSO
- Controlling and monitoring WSim operation
- Running WSim as a permanent task.

It also describes what features are available for operation. For specific information about how to use these features, refer to Part 2, “Operation,” on page 87and *WSim Utilities Guide*.

## Running and analyzing a sample test

Before you run WSim using your complete network definition and all of the message generation decks, you should run one or more sample tests. These sample tests enable you to test portions of the script before you run the complete simulation. You can use the sample tests to find errors in coding or logic before they become masked or compounded by other errors. Sample tests are especially helpful if you are simulating a large network or using complex message generation decks. However, even if you are simulating a small network and using simple decks, dividing the complete test into a number of samples will help you avoid large, time-consuming errors.

In general, your first sample test should include a very small network definition and a simple message generation deck. Subsequent samples could test larger networks or more complicated decks. For example, if you were using WSim to simulate 1000 terminals logging on to TSO, you would run several sample tests. The first test could simulate one terminal and use a simple logon deck. Later tests could use the same message generation deck but would include more terminals. A final sample test might use a more complicated logon deck with error checking.

The number and type of sample tests you run depend on how large your simulated network is and how complicated your message generation scripts are. As you become more experienced with using WSim for a particular type of simulation, you may need to run fewer sample tests. Initially, however, you should plan to conduct a number of sample tests to help you detect and correct errors.

When you run a sample test, you follow the same sequence as you use when you run the complete simulation. This sequence is explained in the following sections of this chapter.

## Preprocessing the script

Preprocessing involves using the Preprocessor to check the syntax of the network definition statements and the message generation decks as well as storing them in the appropriate data sets. Preprocessing is not required; you can store the network definitions and message generation decks in the appropriate data sets using an editor or using the ITPSYSIN program. However, when you preprocess the script, you help ensure that the network will be initialized without errors during WSim operation. Preprocessing does not shorten the amount of time needed to initialize a network, but it may reduce the overall time needed to run WSim by detecting errors early.

If you create your message generation decks using STL, you do not need to use the Preprocessor; the STL translator will check the syntax of your statements and store them in the appropriate data sets. If you create your message generation decks using one of the script generating utilities (WSim/IDC or ITPSGEN) you can preprocess the decks to store them in data sets, but you do not need to check the syntax of the statements. You may want to use ITPSYSIN to store message generation decks created by the script generating utilities without checking syntax. You can preprocess network definitions that correspond to these message generation decks without preprocessing the decks as long as you have already defined the decks and stored them.

## Using the Preprocessor

The input to the Preprocessor consists of the network definition statements, which can be stored as a sequential data set or as a member of a partitioned data set, and one or more message generation decks.

To start the Preprocessor, run the load module ITPENTER with the PREP execution parameter specified. During preprocessing, most of the control blocks that would be needed to run the network are built into virtual storage. Therefore, when you preprocess large network configurations, be sure that the region size is large enough for the network control blocks. For more information about determining region size, refer to “Estimating storage requirements” on page 63.

If the Preprocessor detects no errors in the network definition statements or the message generation decks, the script is stored in the appropriate data sets. You define these data sets in the WSim/ISPF Interface, JCL or TSO CLIST that runs the Preprocessor. You can store the preprocessed network definition statements and message generation decks in the same data set or in separate data sets.

The Preprocessor also provides printed output. This information consists of the following:

- Listing of the input network definition statements (optional)
- Listing of the input errors (if any)
- Network summary report (optional)
- Estimate of storage requirements

  This estimate specifies the number of bytes required for the network control blocks; it does not indicate the region size needed to run WSim for a particular network configuration.
- Notice of whether the network data set was saved in the output data set.

For specific information about using the Preprocessor, refer to *WSim Utilities Guide*, which shows example Preprocessor output with and without errors. It also includes instructions for running the Preprocessor using the WSim/ISPF Interface, sample JCL or a TSO CLIST for running the Preprocessor.

### Using ITPSYSIN

You can use a subset of the Preprocessor functions by running the ITPSYSIN utility program. This utility reads the SYSIN data stream and stores the network definition statements and message generation decks in the appropriate data sets. ITPSYSIN does not check the syntax of the statements.

ITPSYSIN runs much faster than the Preprocessor. You can use ITPSYSIN and then initialize the network using the I (Initialize) operator command with the list option. This will produce a listing of the network definition statements and the message generation decks. By doing this, WSim checks the syntax of the statements only once (when they are initialized) and not 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 processed networks, and automatically generated networks.

**Note:** When you use ITPSYSIN, remember that the syntax of the statements is checked when they are initialized. You must correct all errors before WSim can run.

## Estimating storage requirements

After you preprocessed the script, you can estimate the amount of storage required for the simulation. Calculating the storage needed for large configurations is a complex process because of variations among systems and network requirements. The information presented here is designed to help you estimate the storage required for your particular configuration. These sections discuss estimating host processor virtual storage.

### Host processor virtual storage estimates

The approximate storage requirement, in bytes, for the execution of a simulation is:

Storage Estimate = 1,500,000 + N

where N is the storage required for network definition control blocks.

For an average size network, WSim requires 1,500,000 bytes of virtual storage for the WSim host processor load modules, work areas, and internal buffers. If you plan to simulate a larger network, you need to determine the storage required for network definition control blocks.

#### Network storage size N

You can determine the approximate storage required for network definition control blocks using two methods:

1. Run the Preprocessor, using your network configuration as input. Message ITP657I gives you the storage size required for your network definition. You can then use this value as N.

   This is the recommended method for determining network control block storage requirements. Running the Preprocessor takes less virtual storage than does running the simulation itself because message generation decks and various subtasks are not kept in virtual storage.

   For more information about using the Preprocessor, refer to “Preprocessing the script” on page 61 and to *WSim Utilities Guide*.

2. Consult the example network definition shown in Figure 9.This network contains all of the different types of simulation groups in WSim. It is written in a representative format so that you can estimate how much storage similar groups in your network take.

   You can use this method before you write network definition statements, for example, while you are planning the network. This example only shows samples of the different groups. Your network definitions may or may not require similar storage sizes. In addition, not all definition statements are included in this sample.

## Sample network for estimating storage

The network definition shown in Figure 9 is a sample that contains all groups that WSim can simulate. This network definition can give you an estimate of how much storage may be required to hold the network definition control blocks.

The values used here are components of the variable N. You must add the value N to the other storage estimates to estimate how much virtual storage is required to run the network simulation.

The list below shows the groups that are defined in the sample network and lists their control block storage requirements.

| **Sample Group Type** | |
|---|---|
| (1) NTWRK/PATH | 10,000 |
| (2) APPCLU/TP (CPI-C) | 6,100 |
| (3) VTAMAPPL/LU | 28,400 |
| (4) TCPIP/DEV (Telnet 3270) | 15,800 |
| (5) TCPIP/DEV (FTP) | 6,100 |
| (6) TCPIP/DEV (Simple TCP) | 6,300 |
| **Total N** | **72,700** |

1. These are rounded numbers that represent only the sample network and not all possible configurations.

```
                                                              (1)
STORAGE NTWRK MLOG=YES,INIT=SEC,ITIME=1,CONRATE=YES,         NTWRK/PATH
              UTI=100,MAXSUBA=63,OPTIONS=(CDLOG),            Group
              BUFSIZE=2048,THKTIME=UNLOCK,RESOURCE=ITPECHO
*                                                             Size=10,000
0       PATH WAITDECK
---------------------------------------------------------------------------
                                                              (2)
*
**** ONE VTAM APPC APPLICATION SIMULATING TWO CPI-C TRANSACTION PROGRAMS
*                                                             APPCLU/TP
*                                                             Group
APPCLU1 APPCLU
LU1TPA  TP
LU1TPB  TP                                                    Size=6,100
---------------------------------------------------------------------------
                                                              (3)
*
**** FOUR VTAM APPLICATIONS EACH SIMULATING ONE LU2 HALF-SESSION
*                                                             VTAMAPPL/LU
*                                                             Group
WSIMAPL1 VTAMAPPL
WSIM1    LU  LUTYPE=LU2
WSIMAPL2 VTAMAPPL
WSIM2    LU  LUTYPE=LU2                                       Size=28,400
WSIMAPL3 VTAMAPPL
WSIM3    LU  LUTYPE=LU2
WSIMAPL4 VTAMAPPL
WSIM4    LU  LUTYPE=LU2
---------------------------------------------------------------------------
                                                              (4)
**** ONE TCP/IP CONNECTION SIMULATING FOUR TELNET 3270 SESSIONS
*                                                             TCPIP/DEV
*                                                             Group
TCPTN    TCPIP TCPNAME=TCPIP,SERVADDR=9.67.6.1
TNDEV1   DEV
TNDEV2   DEV                                                  Size=15,800
TNDEV3   DEV
TNDEV4   DEV
---------------------------------------------------------------------------
                                                              (5)
**** ONE TCP/IP CONNECTION SIMULATING FOUR FTP SESSIONS
*                                                             TCPIP/DEV
*                                                             Group
TCPFTP   TCPIP TCPNAME=TCPIP,SERVADDR=9.67.6.2,TYPE=FTP
FTPDEV1  DEV
FTPDEV2  DEV                                                  Size=6,100
FTPDEV3  DEV
FTPDEV4  DEV
---------------------------------------------------------------------------
                                                              (6)
**** ONE TCP/IP CONNECTION SIMULATING FOUR SIMPLE TCP SESSIONS
*                                                             TCPIP/DEV
*                                                             Group
TCPSTCP TCPIP TCPNAME=TCPIP,SERVADDR=9.67.6.3,TYPE=STCP
STCPDEV1 DEV
STCPDEV2 DEV                                                  Size=6,300
STCPDEV3 DEV
STCPDEV4 DEV
---------------------------------------------------------------------------
```

*Figure 9. Sample network for estimating storage*

### Adding terminals

The greatest difference between your network definitions and the example will probably be the number of terminals (DEV, LU, TP) that you define. Some terminal types require more storage than others. In general, 3270 terminals require the greatest amount of storage, followed by LU7s.

When you add 3270 terminals to your network definition, allow at least 3880 bytes of storage for each 3270 DEV or LU2 half-session. The 3270 terminals with additional features and larger display sizes require more storage to simulate.

## Running WSim

You can run WSim on MVS in batch mode, as a procedure, using the WSim/ISPF Interface, or under TSO.

The following sections briefly describe what you need to consider before you run WSim in each of these environments. For specific information about operating WSim, refer to Part 2, “Operation,” on page 87.

### Using MVS

On MVS, you start WSim as a batch job or as a normal procedure. Refer to Part 2, “Operation,” on page 87 for an example of the JCL to run WSim on MVS in each of these configurations.

### Using TSO

You can operate WSim under TSO by starting ITPENTER from a TSO terminal. Operating WSim under TSO can be useful when you do not have access to the system console or when you want WSim operation to be transparent to systems operation personnel. For example, you can operate WSim under TSO when you are conducting a function test on a new application. Note that you probably would not want to operate WSim under TSO when you are conducting a stress or performance test since WSim requires more system resources under TSO.

There are two special installation requirements for operating WSim under TSO: program authorization and the use of the SSP Loader/Dumper utility. Refer to “Authorizing WSim under TSO” on page 10 and the WSim Program Directory for information about these special requirements.

If WSim is going to write its message log data set to a tape data set, you must specify the MOUNT logon attribute for the TSO user ID under which WSim is operating.

Part 2, “Operation,” on page 87 provides more information about operating WSim under TSO. It also includes an example of how you can use a TSO CLIST to allocate WSim data sets and devices, call the WSim load module for execution, and free the data sets and devices.

## Controlling and monitoring WSim operation

The following sections discuss some of the planning considerations for controlling and monitoring WSim operation. They discuss using operator commands, controlling WSim from a console, controlling WSim automatically, and using the Display Monitor Facility. For specific information about how to control and monitor WSim operation, refer to Part 2, “Operation,” on page 87.

## Using operator commands

You can use WSim operator commands to control various aspects of a simulation. For example, you can use operator commands for the following types of tasks:

- Initializing the network
- Starting, stopping, and canceling network resources
- Displaying the current status of a network resource
- Altering the values of network parameters
- Restarting message logging
- Obtaining online response-time statistics for a specified resource
- Entering console recovery mode
- Stopping WSim.

To use operator commands effectively, you need to understand how they affect the resources simulated by WSim.

From the user's standpoint, WSim consists of networks with variable numbers of:

- VTAM applications (VTAMAPPL)
- CPI-C transaction programs (APPCLU)
- TCP/IP resources (TCPIP)

Each resource is represented by a control block. WSim operator commands operate on these control blocks.

You determine which resource is affected by the operator command by specifying a unique resource name on the command. If the name of the resource is not unique (for example, you assigned the same name to terminals on different lines), you must qualify it by preceding it with the name or names of the next higher-level resource or resources. Note that if you assign unique names to the resources in your network, you do not need to qualify the names on the operator command.

You can operate and control networks independently of each other so that you can cancel one network without affecting any other networks. You can also change all networks with one command.

## Controlling WSim from a console

WSim uses the MVS MODIFY command interface to receive WSim operator commands you enter at the operator console. This interface eliminates the need for write-to-operator-with-reply (WTOR) processing by the main WSim simulation task. For more information about using this interface, refer to Part 2, “Operation,” on page 87.

Although you can use the WTOR interface, the system MODIFY command interface is the preferred method of entering commands manually. WSim uses the WTOR interface if WSim is started as a batch job. In this case, a time-stamped outstanding WTOR message is issued to enable you to enter console commands. You must preface each command with the number of the currently outstanding WTOR.

## Controlling WSim automatically

WSim provides several ways for you to control operation automatically, both at the terminal and at the network level. You can do the following to automate WSim operation:

- Use the OPCMND statement to issue operator commands from the message generation deck for a simulated device. You can issue all operator commands using the OPCMND statement, except for console recovery subcommands, which the system console operator must enter. For more information about using operator commands on the OPCMND statement, refer to *Creating WSim Scripts* and *WSim Script Guide and Reference*.
- Use the NTWRK and NTWRKL execution parameters to specify a network to be automatically initialized and started without operator intervention when WSim operation begins. For more information, refer to Part 2, “Operation,” on page 87.
- Code the EMTRATE operand on the NTWRK statement or use the E operand on the A (Alter) operator command to specify an expected message transfer rate for messages transmitted from WSim for the entire network. WSim automatically adjusts the user time interval (UTI) at intervals of specified duration to maintain that rate.

  Once the rate is established, WSim compensates for lines dropping out because of hardware or software problems by lowering the UTI and increasing the rate of traffic from other simulated terminals. For more information, refer to *Creating WSim Scripts* andPart 2, “Operation,” on page 87.
- Code the SCAN operand on the NTWRK statement and use the Q (Query) or A (Alter) operator commands to determine the current status of a simulated terminal or device. You can request that WSim automatically attempt to reactivate the terminal or device if necessary. For more information, refer to *Creating WSim Scripts* and Part 2, “Operation,” on page 87.
- Code the STIME (Start Time) operand on the NTWRK statement to stagger the startup delay for each VTAMAPPL, APPCLU, or TCPIP in the network. Additionally, you can code the FE (Future Event) network definition statement to specify an operator command that is automatically executed when a timer expires or when a specified event occurs. STIME and FE give you control of the entire network, not simply an individual terminal or device. For more information, refer to *Creating WSim Scripts*.
- Use command processors and user exit routines to automate WSim operation.

## Using the Display Monitor Facility

The Display Monitor Facility is a VTAM application program within WSim that can display simulated 3270 display images or display transmitted and received data flows for any simulated device on a VTAM-accesible 3270 monitor. You can use the Display Monitor Facility in the following three ways:

- To develop and debug scripts for display devices
- To monitor a test dynamically during operation
- To show interactions with host applications for user education and product demonstrations.

You can use the Display Monitor Facility with Version 2 Release 1 and later releases of VTAM. The facility runs under MVS. You activate the Display Monitor Facility when you start WSim with the DMAPPL execution parameter specified.

You can use the Display Monitor Function at any time during a simulation. The Display Monitor Facility shows changes to the screen image during the simulation.

### Debugging scripts

The Display Monitor Facility helps speed up WSim script development by enabling you to watch the message traffic being sent and received by a simulated terminal.

This is very useful when you are trying to get a new script to work since you can detect errors as they happen. The Display Monitor Facility helps you detect the following types of errors:

- Invalid cursor position
- Incorrect panel format
- Devices that are no longer generating messages.

### Monitoring WSim

When your scripts are in working order, you can use the Display Monitor Facility to show the current activity of any resource in your simulated network. Without this facility, you would have to use one of the following procedures:

- Issue Q (Query) operator commands to see what was last transmitted or received and how far into the message generation deck the simulation has progressed
- Insert write-to-operator (WTO) statements into your message generation decks or use the STL SAY statement to inform the operator about the progress of the run.

By monitoring the displays during the run with the Display Monitor Facility, the operator console is not cluttered with messages or query responses.

### Demonstrating products

You can use the Display Monitor Facility to demonstrate new products and to help educate users. For example, if you want to demonstrate a new application that uses 3270 terminals, you can code message generation decks to log on and use the application. Then you can use the Display Monitor Facility during a test to show what a real user would see.

For more information about using the Display Monitor Facility, refer to Part 2, “Operation,” on page 87.

## Running WSim as a permanent task

Depending on the method and goals of using WSim in your organization, you may want to run one WSim job as a permanent task. WSim can support multiple network simulations from multiple consoles or can be reused by one person after another. For example, you can run small tests on your production system during normal working hours. You do not need to dedicate a system to the test. Although this method of running WSim would not work for major performance or stress tests, running WSim as a permanent task could ease bottlenecks and scheduling problems for people developing scripts or testing applications.

### Understanding the benefits and concerns

Running WSim as a permanent task can alleviate many organizational and scheduling problems. It can promote the use of WSim for various types of testing. In addition, running WSim as a permanent task can improve the overall quality of testing for the following reason:

- More users will have better access to WSim
- The preparation time for each test will be shorter because all resources are predefined for everyone.

Multiple users can work on WSim concurrently because of the following features:

- Multiple consoles can accept operator commands.

- WTO messages are routed to the console that initialized the network.
- You can use the MODIFY command interface in place of WTOR command input.
- Each network can use a unique log data set.
- The Preprocessor can update definitions while the simulation is running. However, the current simulation will not realize that the network definitions are updated until the network is initialized again.
- The WSim job can use network input definitions from concatenated data sets.
- You can use the O (Output Data) operator command dynamically to close the SYSPRINT output file during the run.
- You can run postprocessor utilities offline with the network log data set previously created.

WSim consumes relatively little host processor time when you are not actively using it. When you want to use WSim, you initialize and start the network. Similarly, another user can initialize and start another network from another console, and so on. When more than one person is using WSim at the same time, you should not enter a ZEND or global C (Cancel) operator command unless all users agree that WSim can be canceled. Instead, you can specify the name of a network on the C operator command and selectively cancel one or more networks without stopping WSim.

## Using the online response-time monitor

Another feature available when you run WSim as a permanent task is a local response-time monitor. One simulated terminal could be running constantly to gather response time statistics from any system in the network. You could then use the RSTATS feature, user-written exit routines, or both to calculate and display response time statistics online. Refer to Chapter 9, “Using WSim output,” on page 71 and to Part 2, “Operation,” on page 87 for more information about using the RSTATS feature. For more information about writing user exit routines, refer to *WSim User Exits*.

You can use the following steps to develop a network that will monitor response times:

1. Define a network containing one VTAMAPPL and one LU statement.
2. Transmit one sample request to the system every few minutes (or at selected time intervals) and measure the time until a response is received. You can perform calculations with RSTATS or with a user exit routine.
3. Maintain running totals and averages internally within WSim by using counters, save areas, and so on.
4. Use the W (RSTATS Query) operator command or a user exit routine to display the response times on a real display terminal.

A response-time monitor can be as simple or complex as you want. By combining scripts, internal logic, and user exit routines, you can locate these custom-made monitors throughout your network with a minimum of overhead.

A sample monitor network, which performs functions other than online response-time measurement, is presented in *Creating WSim Scripts*.

# Chapter 9. Using WSim output

This chapter discusses the planning considerations for generating and using the output from tests. It includes sections on using operator reports, logging messages, formatting the log data set, comparing 3270 display records, and determining response times.

WSim provides a variety of online and printed reports to help you analyze the results of a test. Some of these reports are printed automatically, or you can request them by issuing specific operator commands or running one of the WSim utilities (such as the Loglist Utility or the Log Compare Utility). WSim can provide the following types of output:

- Operator reports that indicate what is happening during operation
- The message log data set that contains complete records of the test run
- Reports generated by the following three utilities to aid you in analyzing the log data set:
  - The Loglist Utility lists the log data set in a formatted report.
  - The Log Compare Utility compares the 3270 display records from two log data sets and reports when a difference is detected.
  - The Response Time Utility provides detailed statistical analysis of the response times.
- Online response-time statistics.

Although WSim output can help you determine the effectiveness of your network or application (for example, by telling you about the response times for a simulated resource), most of the reports are intended to help you understand how WSim is interacting with the system under test.

Remember that WSim is external to the system under test; it acts like an operator typing at a terminal. For this reason, WSim cannot provide information about the internal workings of your system. To obtain this type of information, you should use other monitoring programs during the test. These monitoring programs are the same ones you would use if you were testing your system using real operators.

## Using operator reports

You can use the operator reports to determine what is happening during the test run. This section briefly describes the operator reports you can get during a test. These reports include the following:

- Interval Report
- End of Run Report
- Inactivity Reports.

For general information about the information contained in each these reports, refer to the following sections. For more specific information about these reports, refer to Part 2, “Operation,” on page 87

### Using interval reports

You can use interval reports to monitor what is happening within the simulated network. The Interval Report provides information about the current activity and status of each simulated resource in the network. WSim prints the report on the operator console at the time interval you specify with the ITIME parameter on the NTWRK statement. The statistics are accumulated until you cancel the network with the C operator command or reset it with the R operator command.

Interval reports include the following types of information:
- Time of day and date
- Network name
- Network header
- Value of the network user time interval (UTI)
- Address of port and name of resource
- Current status of resource
- Number of items sent or received by each resource. Depending on the resource, these items can include complete transactions and SNA responses
- Totals of all statistics for each resource for the last reporting interval
- Cumulative totals of all statistics for each resource for all reporting intervals
- Rates per minute of each statistic for the last reporting interval.

You can specify condensed forms of these reports by using the REPORT operand on the NTWRK statement or the X operand on the A (Alter) operator command. For more information about using the REPORT operand, refer to the *WSim Script Guide and Reference* and to *Creating WSim Scripts*.

## Using end of run reports

You can use end of run reports to obtain general information about what happened during the test run. The End of Run Report provides summary data from the simulated network. End of run reports are interval reports that provide you with summary data for the simulated network. They print automatically at the end of each run when you cancel the network or stop WSim. End of run reports have the same format as interval reports.

## Using inactivity reports

The Inactivity Report contains information on the status of devices in the network. You can use this report to find problems in your network. For example, the Inactivity Report can tell you whether a particular terminal is active or inactive.

To obtain this report, code the SCAN operand on the NTWRK statement or use the S operand on the A (Alter) operator command. These operands must specify a time interval between reports that is greater than zero.

The Inactivity Report contains the following information for each inactive resource in the network:
- Last message transmitted
- Last message received
- Time of the last message transmitted
- Time of the last message received
- Name of the message generation deck, if any
- Response field coded on the TEXT statement, if any
- Indication of whether the device was last transmitting or receiving.

The Inactivity Report is time stamped so that you can determine how long a terminal has been inactive. For a list of the criteria the scan function uses to determine if a terminal is active or inactive, refer to *Creating WSim Scripts*.

## Logging messages

When processed by the Loglist Utility, the log data set is probably the single most valuable tool that you can use to debug your network definition statements and message generation decks. The log data set contains all data that has been transmitted or received by the WSim-simulated resources in a specified network. You define the name of the message log data set in the LOGDD DD statement when you run WSim.

By default, the message logging facility is active for the entire network; however, you can deactivate message logging for the entire network or for a line in the network by specifying MLOG=NO on the NTWRK or LINE statement. You can also 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.

In general, you do not need to know how logs and time stamps messages to use the log data set. However, you may find this information helpful as you debug your scripts or analyze response times. For complete information about message logging, refer to Part 2, “Operation,” on page 87 and *WSim Utilities Guide*. For information about formatting the log data set with the Loglist Utility, refer to the following section, Formatting the Log Data Set.

## Formatting the log data set

The Loglist Utility formats and prints the records in the log data set. You can specify the types of records you want to see, and you can limit the resources for which records are printed. This enables you to gain the information that is the most useful to you about the behavior of your network.

In addition, you can write user exit routines to read the log data set and print information about the simulation run. For more information about writing user exit routines, refer to *WSim User Exits*.

### Running the Loglist Utility

There are two types of input to the Loglist Utility: the log data set and the Loglist Utility control commands. The log data set is created automatically when you run WSim, unless you specify MLOG=NO on the NTWRK statement or on all lower-level resources. You can create a file containing control commands that control the operation of the Loglist Utility and format the output produced by the utility. As an alternative, you can specify the CONSOLE execution parameter and enter the control commands directly at the operator console.

You use the WSim/ISPF Interface, JCL or a TSO CLIST to start the utility, to name the input files, and to specify where the formatted log will be printed. For more information about using control commands and running the Loglist Utility, refer to *WSim Utilities Guide*.

### Using the output from the Loglist Utility

The Loglist Utility formats each type of log record differently. You can use the formatted records produced by the Loglist Utility to debug your scripts and to learn about your network. For example, you can determine whether new application program functions ran correctly. You can also obtain an estimate of your system's performance by using the time stamps in each record to compute the response times between the messages transmitted and received by the simulated terminals.

One feature of the Loglist Utility that is especially helpful is the printing of screen image records. These images are updated each time a message is sent or received by the device, and you can tell WSim to log the image each time it is updated by specifying the LOGDSPY operand in the networks.

When the test run is over, you can use the Loglist Utility to format and print these screen images. The output from the Loglist Utility looks the same as the screen images you would see at the real device. You will find it helpful to use this output when you are trying to get a new message generation deck to work because screen image records are easier to understand than the raw 3270 data stream. For specific information about these formatted records, refer to *WSim Utilities Guide*.

## Comparing 3270 display records

The Log Compare Utility compares 3270 display records from two log data sets and reports when a difference is detected. You can select the records to be compared and the fields to be compared for each record. You can also request a set of reports that identify the differences found between the display records.

### Running the Log Compare Utility

There are two types of input to the Log Compare Utility: log data sets from two test runs and the Log Compare Utility control commands. The log data sets are created automatically when you run WSim, unless you specify MLOG=NO and LOGDSPY=NONE on the NTWRK statement. You can create a file that contains the control commands. These commands specify which records from the log data sets to compare and when and how to synchronize the log data sets. As an alternative, you can specify the CONSOLE execution parameter and enter the control commands directly at the operator console.

You use the WSim/ISPF Interface, JCL or a TSO CLIST to start the utility, to name the input files, and to specify where the reports will be printed. For more information about running the Log Compare Utility, refer to *WSim Utilities Guide*.

The control commands for the Log Compare Utility are grouped into two categories:

**Selection commands**
Define the records to be compared.

**Process commands**
Define the fields to be compared on each record. In addition, these commands specify which reports should be generated.

These commands are optional. However, if you enter no control commands except for RUN and END, the Log Compare Utility compares every log display record for every device or logical unit in the log data set. For more information about using control commands for the Log Compare Utility, refer to *WSim Utilities Guide*.

## Synchronizing the log data sets

You should use the synchronization facility provided with the Log Compare Utility to synchronize the two data sets. The synchronization facility helps ensure that WSim is comparing equivalent display records from each of the two data sets. Otherwise, minor changes between the two test runs can significantly affect the results of the Log Compare Utility.

For example, suppose you are using WSim to conduct a regression test on an application. You conduct an initial test, make a few changes to the application (such as adding a password security panel), and repeat the test. You use the Log Compare Utility to find out how the 3270 display records differ between the two test runs and, surprisingly, find significant differences between the log data sets. This is because you added a new panel, so the two log data sets were not synchronized. As a result, the Log Compare Utility attempted to compare two entirely different display records.

Differences between two log data sets can occur for a number of reasons, including the following:

- One or more fields on a panel contains a different value. For example, the value of a particular field may have changed from 45 to 70 between two test runs.
- A new panel is added or deleted. For example, you added a new password security panel between the two test runs.
- A field is added or deleted from a panel. For example, you added a field for keeping track of back orders.

Although these may seem to be minor differences, they can significantly affect the results from the Log Compare Utility. To avoid this problem, you should use the synchronization facility provided with the Log Compare Utility. This facility attempts to resynchronize the compare process if a specified number of data differences occur in succession. To use the facility, you need to identify the following:

- The records that will be used to resynchronize the compare process. For example, if you know that a particular panel has not changed between the two test runs, you can request that the Log Compare Utility use the display record from this panel to resynchronize the comparison, if necessary.
- The number of differences that can occur before the Log Compare Utility ends or resynchronizes the compare process.

For more information about synchronizing two log data sets, refer to *WSim Utilities Guide*.

## Using the output from the Log Compare Utility

Depending on the operands you specify on the REPORT process command, the Log Compare Utility can print one or more of the following reports:

- Active Command List (printed automatically after each run)
- Complete Records List
- Compare List
- Differences Report
- Summary Report.

The following sections describe each of these reports. For more information, refer to *WSim Utilities Guide*.

***Active Command List:*** The Log Compare Utility prints an Active Command List automatically after each run. This report lists the commands you issued and the operand values that were in effect during the run. You can use this report to determine which selection and process commands you used for a particular comparison. For example, if you ran the Log Compare Utility several times, you might use the Active Command List to determine which commands produced which output.

***Complete Records List:*** The Log Compare Utility prints Complete Records Lists for each device being compared. The report lists each display record that was compared as well as those records that were excluded. There are separate lists for each log data set. You request the Complete Records List by specifying the RECORDS operand on the REPORT process command. You can use the Complete Records List to verify that the Log Compare Utility attempted to compare the display records you specified.

***Compare List:*** The Log Compare Utility prints one Compare List for each device being compared. The report lists the following items:
- Each pair of display records (one from each log data set) that was compared
- Any process commands used to process the records
- The results of the comparison. If a difference was detected, the Compare List includes a reason for the difference.

You request the Compare List by specifying the COMPARES operand on the REPORT process command. You can use this report to determine quickly whether the comparison occurred correctly; a quick scan of this report indicates whether synchronization was attempted and so on.

***Differences Report:*** If differences were detected, the Log Compare Utility prints one Differences Report for each device being compared. The information contained in the report depends on whether the difference was a data difference or an attribute difference. For example, if the Log Compare Utility detects a data difference, it prints a copy of the formatted screen image for both data sets. If the Log Compare Utility detects an attribute difference, it prints a detailed attribute report. You request the Differences Report by specifying the DIFFERENCES operand on the REPORT process command. This report is printed by default if you do not specify the REPORT command. You can use this report when two screens do not match and you are not sure why.

***Summary Report:*** The Log Compare Utility prints one Summary Report following each run. This report summarizes the overall results of the run. It includes the following information for each device:
- Number of records selected
- Number of records excluded
- Number of differences detected
- Whether synchronization was attempted
- Whether the run was ended early.

You request the Summary Report by specifying the SUMMARY operand when you specify the REPORT process command. This report is printed by default if you do not specify the REPORT command. You can use this report as a quick reference after the comparison is completed.

## Determining response times

WSim provides the following two facilities for monitoring response times:

**Response Time Utility**
A postprocessor that uses data from the log data set.

**RSTATS**
An online response statistics reporting facility. RSTATS is unrelated to the Response Time Utility and enables you to monitor the response times of simulated devices while WSim is running.

For more information about using the Response Time Utility, refer to Using the Response Time Utility and *WSim Utilities Guide*. For more information about using RSTATS, refer to "Using the response-time statistics feature" on page 79 and Part 2, "Operation," on page 87.

## Using the Response Time Utility

The Response Time Utility analyzes the log data set and measures the time it takes to enter a command at a simulated terminal and receive a response from the system under test. To do this, the utility uses the time stamps from a pair of transmit and receive records and calculates response times for each terminal. 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 a logical transaction for the utility to use when it selects the transmit and receive records. This logical transaction can include any number of data exchanges between the simulated resource and the system under test.

**Note:** Because the Response Time Utility determines the transmit-receive pairs for a particular terminal, be sure to use unique names for the terminals in your network definition if you plan to use this utility.

### Running the Response Time Utility

There are two types of input to the Response Time Utility: the log data set and the Response Time Utility control commands. The log data set is created automatically when you run WSim, unless you specify MLOG=NO on the NTWRK statement. You can create the file that contains the control commands. These commands specify what time limits to use, which lines and terminals to evaluate, how to compute response times, and how to generate output reports. As an alternative, you can specify the CONSOLE execution parameter and enter the control commands directly at the operator console.

You use the WSim/ISPF Interface, JCL or a TSO CLIST to start the utility, to name the input files, and to specify where the reports will be printed. For more information about running the Response Time Utility, refer to *WSim Utilities Guide*.

### Defining transactions

You can define logical transactions to the Response Time Utility by specifying the messages that mark the beginning and end of the transaction. These transactions can include any number of messages. For more information about transactions, refer to "Deciding which transactions to test" on page 46.

For more information about defining transactions to the Response Time Utility, refer to *WSim Utilities Guide*.

## Using output from the Response Time Utility

The Response Time Utility reads the records on the log data set and calculates response times based on the data. Depending on the commands you specify, the Response Time Utility can print the following reports:

- Response Time Reports
- Transaction Record Listing
- Response Listing File
- Response Time Frequency Distribution
- Cumulative Response Time Distribution
- Time Graph of Responses.

This section briefly describes each of these reports. For more information, refer to *WSim Utilities Guide*.

***Response Time Reports:*** Response Time Reports include comprehensive summary and statistical information about the response times for terminals or user-defined transactions. You request these reports by specifying the REPORT control command and the REPORT operand on the TERM, VTAMAPPL, or APPCLU control commands.

***Transaction Record Listing:*** The Transaction Record Listing lists the log records that were selected for transaction processing by the other input commands. No records are listed unless you defined at least one transaction. You can use this report to verify that the response-time calculations were made using the records you expected to be used.

You request this listing by specifying the TPRINT control command.

***Response Listing File:*** The Response Time Utility can create a sequential data set that consists of one record for each response time computed during the run. Each record consists of information that identifies the terminal or transaction for which the response was computed and the response time. To obtain the Response Listing File, you must do the following when you run the Response Time Utility:

- Name the file LISTDD in the JCL statement.
- Specify the LIST or LISTX execution parameter.

If you specify LISTX, the Response Time Utility produces additional information that can be processed by the Service Level Reporter (SLR) licensed program using user-defined SLR Adaptable Log Layout (ALL) log tables and user programs. SLR can create color graphs and charts from the response time data. For more information about the format of the extended records produced when you specify the LISTX execution parameter, refer to *WSim Utilities Guide*.

***Response Time Frequency Distribution:*** The Response Time Frequency Distribution is a histogram that shows for each calculated response time, what percentage of responses fall into that response time. You request the distribution by using the REPORT control command and the REPORT operand on the VTAMAPPL, APPCLU, or TERM commands.

***Cumulative Response Time Distribution:*** The Cumulative Response Time Distribution is a graph that shows the cumulative distribution of response times for the time period defined for the run. You request the distribution by using the REPORT control command and the REPORT operand on the VTAMAPPL, APPCLU, or 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 time. Each increment on the horizontal axis represents a time interval that you specify. The graph plots the minimum, average, and maximum response time computed during that interval. You request a time graph by using the REPORT control command and the REPORT operand on the VTAMAPPL, APPCLU, or TERM commands.

## Using the response-time statistics feature

You can use the Response-Time Statistics feature, RSTATS, to provide online response time calculations for terminals simulated by WSim. RSTATS measures the time it takes to enter a command at the terminal and receive a response from the system under test.

RSTATS collects online response-time statistics only for those simulated resources that generate messages. These include terminals and logical units. WSim supports the RSTATS feature for all terminal types.

### Deciding whether to use RSTATS or the Response Time Utility

In most cases, the results from RSTATS should closely resemble the results from the Response Time Utility because RSTATS processes the same LOG control block immediately after it is written to the log data set. However, RSTATS calculates response times during operation, which prohibits some of the more complex processing available when you use the Response Time Utility. Like the Response Time Utility, RSTATS processes every transmit and receives record. However, when you use RSTATS, you cannot define logical transactions. In addition, certain messages are screened and discarded for SNA, 3270, and LU7 devices.

In general, you use the RSTATS feature when you want to monitor response times during operation to ensure that messages are being transmitted and received at reasonable rates. You use the Response Time Utility when you need more detailed statistics about response times after a test run.

### Activating the RSTATS feature

To activate the RSTATS feature for a particular terminal, device, or logical unit, you must do the following:

1. Code RSTATS=YES on the appropriate DEV or LU statement or statements when you define the network.
2. Specify the name of the terminal, device, or logical unit on the W (RSTATS Query) operator command to generate the statistics.

You can reset the RSTATS feature for a particular resource by using the A (Alter) operator command or for the entire network by using the R (Reset) operator command. For more information about using these operator commands, refer to Part 2. Part 2, “Operation,” on page 87. For more information about coding the RSTATS operand, refer to the *WSim Script Guide and Reference*.

### Using the output from RSTATS

RSTATS provides the following information for each device or logical unit:

- Average response time
- Most recent value of response time
- Lowest value of response time
- Highest value of response time
- Total number of responses or completed transactions.

You can use this information to help ensure that your simulation is proceeding correctly. For more information about using RSTATS, refer to Part 2. Part 2, "Operation," on page 87.

# Chapter 10. Sample files

The WSim installation tape contains a sample data set. This data set contains the following members that may be useful when installing WSim or writing exit routines.

## MVS sample data set

| | |
|---|---|
| **ACHKNETV** | Message generation deck used by AVMON LU to monitor NetView. |
| **ACHKTSO** | Message generation deck used by AVMON LU to monitor TSO. |
| **ACTRLNET** | Message generation deck used by AVMON network controller LU. |
| **AFORTIME** | AVMON message generation deck that reformats the time of day. |
| **ALOGNETV** | Message generation deck used by AVMON LU to log on NetView. |
| **ALOGTSO** | Message generation deck used by AVMON LU to log on TSO. |
| **AMONNETV** | Message generation deck used by the NetView subsystem controller LUs in AVMON network. |
| **AMONTSO** | Message generation deck used by the TSO subsystem controller LUs in AVMON network. |
| **AVMON** | Network definition statements for the AVMON network. |
| **COMPJOB** | Sample Log Compare Utility JCL. |
| **CPICCON** | CPI-C constants STL include file. |
| **CPICVAR** | CPI-C variables STL include file. |
| **ECHO** | Sample PLU echo network. |
| **ECHOJOB** | Sample ITPECHO JCL. MFS file used to define the Balance Inquiry Screen in the WSim/ATP example. |
| **INSTALL1** | Sample installation test network 1. |
| **ITMNUSER** | New user setup JCL for WSim Test Manager. |
| **ITPACCEP** | Job stream to ACCEPT the WSim product using SMP/E. |
| **ITPALLOC** | Job stream to allocate the WSim and distribution libraries for SMP/E installation. |
| **ITPAPPLY** | Job stream to APPLY the WSim product using SMP/E. |
| **ITPDDDEF** | Job stream to create DD definitions (DDDEFs) for SMP/E installation. |
| **ITPDFCSI** | Job stream to allocate SMP/E data set and initialize the global zone. |
| **ITPDFZON** | Job stream to define new SMP/E target and distribution zones. |
| **ITPRECEV** | Job stream to RECEIVE the WSim product using SMP/E. |
| **ITPRATEG** | FORTRAN source for the rate table generator. |
| **LLJOB1** | Sample Loglist Utility JCL. |
| **LLJOB2** | Sample Loglist Utility JCL. |
| **LU2RFJOB** | Sample NPM/LU2 Reformatter Utility JCL. |
| **PREPJOB** | Sample Preprocessor JCL. |
| **RESPJOB1** | Sample Response Time Utility JCL. |
| **RESPJOB2** | Sample Response Time Utility JCL. |

| | |
|---|---|
| **SGENJOB** | Sample Script Generator Utility JCL. |
| **SRATETBL** | Sample Rate Tables. |
| **STCPUDP** | Sample Simple TCP and UDP Client network. |
| **STCPUDPS** | Sample STL program for Simple TCP and UDP clients. |
| **STLAVMON** | Sample STL program with AVMON procedure. |
| **STLECHO** | Sample STL program which may be used with ECHO network. |
| **STLINST** | Sample STL program which may be used with the INSTALL1 network. |
| **STLJOB** | Sample STL Translator JCL. |
| **SYSINJOB** | Sample ITPSYSIN JCL. |
| **TAPING** | Sample CPI-C network defining a WSim client. |
| **TAPINGD** | Sample CPI-C network defining a WSim server. |
| **VTBRFJOB** | Sample ITPVTBRF JCL. |
| **WEBLOAD** | Sample network to simulate WEB clients. |
| **WSIMNET1** | Sample WSim network. |
| **WSIMPRC5** | Sample execution JCL. |
| **WSIMPRC6** | Sample execution JCL. |

## CLIST data sets

| | |
|---|---|
| **ECHORUN** | Sample ITPECHO CLIST. |
| **IDC** | Sample Interactive Data Capture Utility CLIST. |
| **IDCTSO** | Sample Interactive Data Capture Utility CLIST, run under TSO using the TSO console option. |
| **LOGCOMP** | Sample Log Compare Utility CLIST. |
| **LOGLIST** | Sample Loglist Utility CLIST. |
| **LU2RF** | Sample SNA 3270 Reformatter Utility CLIST. |
| **PREP** | Sample Preprocessor CLIST. |
| **RESPTIME** | Sample Response Time Utility CLIST. |
| **SGEN** | Sample Script Generator Utility CLIST. |
| **STL** | Sample STL Translator CLIST. |
| **SYSIN** | Sample ITPSYSIN CLIST. |
| **WSIMAPPL** | Sample execution CLIST for VTAMAPPL, CPI-C, or TCP/IP network. |
| **WSIMRUN** | Sample execution CLIST. |
| **WSIMVTAM** | Sample execution CLIST for VTAMAPPL network. |

# Chapter 11. Using WSim to measure response times

This chapter describes the features that enable WSim to measure system response times.

One of the most important considerations in a communication network is the system response time. You often want to monitor response times and to keep them under certain thresholds set by your local computing center staff. You can use WSim to measure and report response time statistics. The following sections discuss ways to do this.

## RSTATS feature

WSim can monitor response time statistics and report them online. You can use the RSTATS (Response Statistics) feature for any simulated terminal including non-SNA devices. WSim measures terminal response times for every transmit-receive pair that is processed. You can display these statistics at any time during the run by using the W (RSTATS Query) operator command. Included in the data are the average response time, lowest and highest values, and last response time calculated. Refer to "Using the response-time statistics feature" on page 79 and Part 2. Part 2, "Operation," on page 87 for more information about RSTATS.

## Response Time Utility

The Response Time Utility provides response time statistics and graphs for selected terminals or the entire simulated network after the simulation is finished. It provides you with a great deal of flexibility in defining the beginning and end of logical transactions, so that you can measure particular responses or track system response statistics over time. Refer to "Using the Response Time Utility" on page 77and to *WSim Utilities Guide* for more information about the Response Time Utility.

# Chapter 12. Summary of logical unit (LU) types

This chapter describes the logical unit (LU) types that WSim currently supports. It also gives an example of the hardware or software products that typically use each type of logical unit. For more information about these LU types, refer to *Systems Network Architecture: Reference Summary*.

**LU Type 0**
LU Type 0 uses SNA-defined protocols for transmission control and data flow control but uses user-defined or implementation-defined protocols to supplement or replace higher-layer protocols. For example, use an LU Type 0 for applications using IMS/VS to communicate with an IBM 4700 Finance Communication System.

**LU Type 1**
LU Type 1 is used for communication with single-device or multiple-device data processing workstations in the following environments:

- Interactive
- Batch
- Distributed data processing.

This type of logical unit uses the SNA character string (SCS) or the Document Content Architecture (DCA) data stream. For example, use an LU Type 1 for applications using IMS/VS and communicating with an IBM 8100 Information System.

**LU Type 2**
An LU Type 2 is for an application that communicates with a single display workstation in an interactive environment. It uses the SNA 3270 data stream. For example, use an LU Type 2 for applications using IMS/VS or CICS/VS and communicating with an IBM 3179 Color Display Station.

**LU Type 3**
LU Type 3 is used for communication with a single printer. It uses the SNA 3270 data stream. For example, use an LU Type 3 for applications using CICS/VS and sending data to an IBM 3287 Printer attached to an IBM 3274 Control Unit.

**LU Type 4**
LU Type 4 is used for either of the following:

- Communication with a single-device or multiple-device data processing or word processing workstation in an interactive, batch, or distributed data processing environment.

  For example, use an LU Type 4 for an application that uses CICS/VS and communicates with an IBM 6670 Information Distributor.
- Logical units in peripheral nodes that communicate with each other.

  For example, use an LU Type 4 for an IBM 6670 Information Distributor communicating with another 6670.

An LU Type 4 uses SNA character strings (SCS) for data processing environments and Office Information Interchange (OII) Level 2 for word processing environments.

**LU Type 6.1**
LU Type 6.1 is used for an application subsystem that communicates with

another application subsystem in a distributed data processing environment. For example, use an LU Type 6.1 for an application program using CICS/VS to communicate with an application program using IMS/VS.

**LU Type 6.2**

LU Type 6.2 is used for an application that communicates with another application in a distributed data processing environment. It uses either of the following data streams:

- SNA general data stream (GDS)
- User-defined data stream.

LU Type 6.2 sessions provide communication between two Type 5 nodes, a Type 5 node and a Type 2.1 node, or two Type 2.1 nodes. Examples of an LU Type 6.2 include the following:

- An application that uses CICS/VS and communicates with another application program that uses CICS/VS
- A CPI-C client transaction program that runs on a workstation and communicates with a CPI-C server running on a CICS/VS host system
- An application in a System/38 that communicates with an application in a System/36.

**LU Type 7**

An LU Type 7 is for an application that communicates with a single display workstation in an interactive environment. It uses the 5250 data stream. An example of this type of logical unit is an application running on a System/34 that communicates with an IBM 5251 Display Station.

Below illustrates which LU types can be used with different types of simulations.

| **Simulation Types** | **LU Types** |
|---|---|
| VTAMAPPL | All |
| SNA | All |

# Part 2. Operation

# Chapter 13. Introduction to WSim operation

The remaining chapters of this book describe the following aspects of WSim operation:

- Operating WSim on MVS with JCL and TSO CLISTs

  On MVS, you can run WSim as a batch job or started procedure from the system console or from a TSO ID as a CLIST. Running WSim under TSO may be preferable when multiple users are running WSim simulations simultaneously or when access to the system console is not convenient, but may be limited by hardware constraints, such as communication controller equipment, depending on the configurations being simulated. For large stress runs, it is recommended that WSim be run from the system console, rather than under TSO to reduce the amount of system resources used by WSim.
- Entering operator commands to control WSim

  After you start WSim, you can control the simulation run by entering operator commands. For example, you can use operator commands to:
  - Start and stop networks or specific network resources
  - Change network parameters, such as user time intervals, message logging, intermessage delays, and report intervals
  - Signal events
  - Query network resources
  - Recover hung terminals.
- Formatting and printing reports provided by WSim

  WSim provides several operator reports that provide information about the simulation run. This information is formatted and written to the printer designated by the JCL or CLIST used to run WSim. These reports include:
  - Interval reports, which provide counts of message traffic for the network resources at user specified intervals.
  - End of run reports, which summarize the message traffic data for the entire simulation run.
  - Inactivity reports, which provide information about terminals which have not sent or received any data during a user specified interval.
- Logging messages, including how to control, route, inhibit, and restart message logging with operator commands

  WSim provides the capability to log all messages sent and received by the simulated network. On MVS, the message log data can be written to a data set (DASD or tape).
- Monitoring WSim with the Display Monitor Facility

  With the Display Monitor Facility, you can actually see the formatted screens and data being sent and received by a simulated 3270 terminal on a real 3270 terminal. For non-3270 simulated devices, you can display the actual data streams. This facility is extremely useful during the development of scripts, as well as a real-time monitor of terminal activity.
- Classifying, isolating, and reporting problems

  WSim is typically used in a very complex environment involving many layers of interacting hardware and software. An error in the setup or operation of any of these layers can cause problems during a simulation run. You should become familiar with some of the more common user errors that can occur and how to isolate them.

In the event that you should encounter a problem in WSim itself, there are recommendations for the types of problem documentation needed to diagnose and correct the problem.

# Chapter 14. Running WSim

You can run WSim on MVS (either in batch mode, as a procedure, under TSO, or under ISPF). For information about the execution parameters you can use to run WSim, refer to "Using WSim execution parameters" on page 95.

## Running WSim on MVS with JCL

The following sections discuss how to run WSim on MVS using JCL. They describe the JCL needed to run WSim and discuss dynamic allocation of the SYSPRINT data set.

### Using JCL for CPI-C, VTAMAPPL or TCP/IP simulations

The example below shows the JCL you can use to run WSim for simulations of CPI-C transaction programs or VTAM applications, or for TCP/IP simulations. When running a CPI-C transaction program or VTAM application simulation, WSim communicates either with a CPI-C transaction program or a VTAM application within the same host or in another host. In the latter case, VTAM will route traffic to the other host by means of a live SNA network. For a TCP/IP simulation, WSim communicates to the TCP/IP virtual machine or address space on the same host.

Use this JCL if you want to log messages on a tape. This can be found in the WSIM.SITPSAMP data set as member WSIMPRC5.

```
//WSIMPRC5 PROC
//GO       EXEC PGM=ITPENTER,REGION=2048K
//*OPTIONAL EXECUTION PARAMETER DATA SET
//PARMDD   DD   DSN=WSIM.PARMS,DISP=SHR
//STEPLIB  DD   DSN=WSIM.SITPLOAD,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//INITDD   DD   DSN=WSIM.TESTFILE,DISP=SHR
//MSGDD    DD   DSN=WSIM.MSGFILE,DISP=SHR
//* OPTIONAL, REQUIRED ONLY WHEN USING RATE TABLES
//RATEDD   DD   DSN=WSIM.SITPRTBL,DISP=SHR
//*
//*             MESSAGE LOGGING OUTPUT DATA SET ON A TAPE
//LOGDD    DD   DSN=WSIM.MSGLOG,DISP=(NEW,KEEP),UNIT=TAPE,
//             LABEL=(,NL),VOL=(,,,20)
//*
//*              OPTIONAL NETWORK LOG DATA SET
//NTWRKLOG DD   DSN=WSIM.NETLOG,DISP=(NEW,KEEP),UNIT=TAPE,
//             LABEL=(,NL),VOL=(,,,20)
//*
//*              OPTIONAL MESSAGE TEXT WORK DATA SET
//MSGDISK  DD    UNIT=SYSDA,SPACE=(4096,100)
```

Use the example JCL, shown below, if you want to log messages on a disk. This can be found in the WSIM.SITPSAMP data set as member WSIMPRC6.

```
//WSIMPRC6 PROC
//GO       EXEC PGM=ITPENTER,REGION=2048K
//*OPTIONAL EXECUTION PARAMETER DATA SET
//PARMDD   DD   DSN=WSIM.PARMS,DISP=SHR
//STEPLIB  DD   DSN=WSIM.SITPLOAD,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//INITDD   DD   DSN=WSIM.TESTFILE,DISP=SHR
//MSGDD    DD   DSN=WSIM.MSGFILE,DISP=SHR
//* OPTIONAL,  REQUIRED ONLY WHEN USING RATE TABLES
```

```
//RATEDD   DD   DSN=WSIM.SITPRTBL,DISP=SHR
//*             MESSAGE LOGGING OUTPUT DATA SET ON A DISK
//LOGDD    DD   DSN=WSIM.MSGLOG,DISP=(NEW,KEEP),UNIT=3380,
//             VOL=SER=YOURPK,SPACE=(CYL,(10,10))
//*              OPTIONAL NETWORK LOG DATA SET
//NTWRKLOG DD   DSN=WSIM.NETLOG,DISP=(NEW,KEEP),UNIT=3380,
//             VOL=SER=YOURPK,SPACE=(CYL,(10,10))
//*              OPTIONAL MESSAGE TEXT WORK DATA SET
//MSGDISK  DD    UNIT=SYSDA,SPACE=(4096,100)
```

For more information about simulation of CPI-C transaction programs, VTAM application programs, or TCP/IP client applications, refer to Part 1. Part 1, "Planning and installation," on page 1 and *Creating WSim Scripts*.

## Description of the JCL data sets

Below describes the functions of each of the JCL statements used in the examples shown in this chapter.

| Statement | Function |
|---|---|
| **PROC** | Initiates the procedure. |
| **EXEC** | Specifies the program name and any optional execution parameters. The parameters are discussed in "Using WSim execution parameters" on page 95. |
| **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 one 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. |
| **STEPLIB DD** | Defines one or more data sets containing the host processor modules. |
| **SYSPRINT DD** | Defines the output printer. This statement is optional when running on MVS. (See Notes No.1) |
| **INITDD DD** | Defines a partitioned data set containing the network definition statements. The BLKSIZE for this data set must be a multiple of 80, and it must use a fixed block record format (RECFM=FB). This data set can be the same as the MSGDD data set. |
| **MSGDD DD** | Defines a partitioned data set containing the message definition decks. The BLKSIZE for this data set must be a multiple of 80, and it must use a fixed block record format (RECFM=FB). This data set can be the same as the INITDD data set. |
| **RATEDD DD** | Defines a partitioned data set containing the rate table members. This statement is required only when using rate tables. |
| **LOGDD DD** | Defines the sequential output data set used for message logging. This statement is required when MLOG=YES is coded in the network. (See Notes No.2) |
| **NTWRKLOG DD** | Defines a separate sequential log data set for the network. You can use operands on the NTWRKLOG statement to specify your own DD statement names to replace NTWRKLOG DD. |

| | |
|---|---|
| **MSGDISK DD** | Defines the sequential work data set for paging MSGTXT, user table, and IF statement control block data. You can use operands on the MSGDISK statement to specify your own DD statement names to replace MSGDISK DD. (Refer to the MSGDISK statement description in *WSim Script Guide and Reference* for more information.) |

**Notes:**

- The SYSPRINT data set is dynamically allocated if the SYSPRINT DD statement is omitted in the JCL. Use the O (Output Data) operator command to release any data accumulated so far. Otherwise, all SYSPRINT output is held until the job ends, unless you directly allocated SYSPRINT data to a real printer. Refer to "Allocating the SYSPRINT data set" for more information.
- Change the volume and unit parameters on the LOGDD DD or NTWRKLOG DD statement if you want to allocate more than one tape drive. The LOGDD DD or NTWRKLOG DD statement has a BLKSIZE default and minimum value of 8160, and a default value of 5 for the number of channel programs (NCP). If you experience lost message log data (indicated by error message ITP030I), you may increase these two values. For example, to allocate 10 buffers, each having 23476 bytes of virtual storage, for message logging, you could specify the parameters DCB=(BLKSIZE=23476,NCP=10).

  Note that when running the postprocessors, the BLKSIZE parameter on the SYSUT1 DD statement (if specified) must match or exceed what was specified on the LOGDD DD statement of the simulation run.
- When running under TSO, the BLKSIZE and NCP parameters can only be specified for NEW or MOD data sets.
- The maximum value for BLKSIZE is 32760. The maximum recommended value for a 3350 disk is 19069. The maximum recommended value for a 3380 disk is 23476. The maximum recommended value for a 3390 disk is 27998.
- When using a DASD for the message-logging output data set, the SPACE parameter must define a data set large enough to contain all message-log data generated. The DASD used must be capable of recording blocks 8160 bytes long.

## Allocating the SYSPRINT data set

When WSim executes ITPENTER on MVS, it attempts to dynamically allocate SYSPRINT as a SYSOUT data set. Allocating SYSPRINT lets WSim produce online output reports without dedicating a real printer to WSim. The output class defaults to the MSGCLASS of the job, and the space allocated is the system default. If you want to change the class, space, or some other parameter, you can use the superzap service aid available with MVS to modify the entries in CSECT ITPS99TU in module ITPS99TU. The module format is described in the next section.

If you include the SYSPRINT DD statement in the JCL you use to run WSim, the dynamic allocation fails and the SYSPRINT output is handled in the standard way. In order to use dynamic allocation of the SYSPRINT data set, you must omit the SYSPRINT DD statement from the JCL.

When you use dynamic allocation, all data written to SYSPRINT can be printed by the MVS system writer routines without affecting operation. To release the spooled output, enter the O (Output) operator command without operands. The O command causes the SYSPRINT data set to be closed, freed (since it is allocated with the FREE=CLOSE attribute), reallocated, and reopened.

### Formatting ITPS99TU

The ITPS99TU module is made up of 10 fullword pointers that address text units, followed by the 10 text units used in dynamic allocation. The first three text units specify the DDNAME, SYSOUT, and FREE=CLOSE parameters and should not be changed. You can use the last seven text units to modify the default class, space allocation, output form, and other parameters. Each of the last seven text units is 14 bytes long, and all bytes are initialized to binary zeros. The fourth text unit (the first one available for modifying a parameter) is located at offset X'52' in the CSECT ITPS99TU. The seven text units for modifying parameters are located at hexadecimal offsets 52, 60, 6E, 7C, 8A, 98, and A6.

You can find information about the formats of the various text units for data set name (DSNAME) allocation in *MVS/ESA System Programming Library: Application Development Guide*, GC28-1852.

## Running WSim on MVS with a TSO CLIST

You can run WSim under the control of TSO by starting ITPENTER from a TSO terminal. Running WSim under TSO can be useful in situations where you do not have access to the system console or where you want operation to be transparent to system operation personnel. For example, you can run WSim under TSO when you are conducting function tests on new applications. But, you would probably not want to run WSim under TSO when you are conducting a stress test because WSim requires more system resources under TSO.

### Planning considerations

There are two special installation requirements for running WSim under TSO: program authorization and the use of the SSP Loader/Dump utility. Refer to Part 1, “Planning and installation,” on page 1 and *WSim Program Directory* for information about planning.

If WSim is going to write its message-log data set on a tape, the TSO user ID under which WSim is running must have the MOUNT logon attribute specified.

### Communicating with a TSO terminal

When running under TSO, WSim issues TGET and TPUT macros to allow communication with the TSO terminal operator. All operator messages are written to the TSO user's terminal with a TPUT macro. Once each second, a TGET macro with the NOWAIT option is issued to retrieve any TSO terminal operator input. If messages being written to the TSO terminal are overwriting the input before it can be sent, the terminal operator can issue a null message (the Enter key alone); WSim will write the message ITP001E and issue a TGET with the WAIT option. This suspends output until the operator enters the input.

### Using a TSO CLIST

The example below shows how you can use a TSO CLIST to allocate data sets and devices, call the WSim load module for execution, and free the data sets and devices. See “Description of the JCL data sets” on page 92 for a description of the CLIST data sets.

**Note:** STEPLIB should be pointing to WSIM.SITPLOAD through the user logon procedure. You should also include a SYSDUMP statement in the user logon procedure.

This can be found in the WSIM.CLIST data set as member WSIMRUN.

```
ALLOC DDNAME(SYSPRINT) UNIT(E) NEW
ALLOC DDNAME(PARMDD) DSNAME('WSIM.PARMS') SHR
ALLOC DDNAME(INITDD) DSNAME('WSIM.TESTFILE') SHR
ALLOC DDNAME(MSGDD) DSNAME('WSIM.MSGFILE') SHR
ALLOC DDNAME(RATEDD) DSNAME('WSIM.SITPRTBL') SHR
ALLOC DDNAME(LOGDD) NEW UNIT(TAPE) VOLUME(WSIMTP)-
      LABEL(NL)
CALL 'WSIM.SITPLOAD(ITPENTER)'
FREE DDNAME(SYSPRINT)
FREE DDNAME(INITDD)
FREE DDNAME(MSGDD)
FREE DDNAME(RATEDD)
FREE DDNAME(LOGDD)
```

**Notes:**

- If you are using a user exit program, concatenate the data set that contains the user exit module to WSIM.SITPLOAD in the TSO logon procedure.
- The BLKSIZE and NCP parameters can only be specified for NEW or MOD data sets. If you are logging messages to disk and specifying BLKSIZE and NCP, your ALLOC statement should look something like the following:

```
ALLOC DDNAME(LOGDD) MOD DSNAME('WSIM.LOG') BLKSIZE(23734) -
      NCP(5)
```

## Running WSim under ISPF

You can also run WSim under ISPF by invoking the WSim/ISPF Interface. To do this, follow these steps:

1. Invoke the WSim/ISPF Interface main panel from ISPF[2]. 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 Run a Simulation panel is displayed.

   **Note:** You can also type RUNWSIM on the WSim/ISPF Interface main panel command line and press **Enter** to display this panel.
3. Fill in the appropriate fields.
4. Select the appropriate configurations by typing a / or s next to the selection field. You can select any combination of configurations that you want to run.
5. Press **Enter**. Depending on the configuration you selected, the appropriate full-screen pop-up panel is displayed. Fill in the appropriate information on this panel and press **Enter**.

For more information on the WSim/ISPF Interface, refer to *WSim Utilities Guide*.

## Using WSim execution parameters

This section describes the execution parameters that you can use when running WSim. You can include execution parameters on the EXEC statement (MVS) or in a data set defined on the PARMDD statement. Parameters in the PARMDD data set are processed first. When you run WSim as a started procedure, you can specify

2. Alternatively the WSIMISPF exec can be used to start the interface. This exec starts the WSim/ISPF Interface without having the data sets concatenated in the TSO logon procedure.

execution parameters on the MSV START operator command. The parameters you specify override any parameters in the PARMDD data set or on the EXEC statement. All parameters are optional.

**CPUSTATS**
: Specifies that host-processor wait time statistics are collected for WSim. Message ITP138I is written to the operator console for the main simulation task for each minute of the simulation run. The message will give the percentage of the previous minute that the task was not waiting for work to be done.

**DMUSERPW=***cpssword*
: Specifies the password you must enter after logging onto the Display Monitor Facility, where cpssword is a 1- to 8-character name. If DMUSERPW is not specified, no password entry is required after logging onto the Display Monitor Facility.

**DMAPPL**=*applid*
: Specifies the name of the APPL defined to VTAM for Display Monitor Facility sessions.

**NCP**=*n*
: Specifies the default number of buffers allocated for output to each log data set. You can override this value with the NCP suboperand of the DCB operand on the DD statement for the log data set (MVS only) or with the NCP operand on the NTWRKLOG statement for individual network log data sets. The default NCP value is 5. The maximum NCP value allowed is 255; the minimum is 2. If 0 or 1 is specified, 2 will be used.

**NOWTOR**
: Allows WSim to be executed under MVS as a batch job without issuing a WTOR for operator control. When NOWTOR is specified, no WTOR will be issued and WSim must either be ZENDed by an operator command from an active network or canceled by the operator.

**NTWRK=***netname*
: Specifies the name of a network that is to be automatically initialized and started without operator intervention. *netname* is the name of the INITDD data set member containing the wanted network definition.

**NTWRKL=***netname*
: Performs the same function as the NTWRK parameter, but also specifies that the network will be listed on the SYSPRINT data set as it is initialized.

**PRTLNCNT=***nnn*
: Specifies the maximum number of lines to be printed on a page of 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 message routing codes to be used in writing messages to the operator. Each n specifies a system routing code that defines a console destination for all write-to-operator (WTO) and write-to-operator-with-reply (WTOR) messages written by WSim. n is an integer from 1 to 16. The default is 8 if WSim is started from a batch job. By default, if WSim is started as a procedure, the messages will be routed to the console that started the procedure or to the console at which the operator command was entered.

  This parameter is meaningful only in an MVS environment outside of TSO.

## Understanding return codes

When WSim ends, a return code is set to indicate the status from the operation. WSim can return the following codes:

| Code | Meaning |
|---|---|
| **0** | Run complete with no errors. |
| **8** | Required initiator data set could not be opened. |
| **12** | Invalid EXEC parameter specified. |

# Chapter 15. Using operator commands

This chapter describes how to use operator commands to control a simulation.

## Introducing operator commands

You can use operator commands to control various aspects of a simulation. You usually enter these commands from a console, but you can also generate them automatically by using the OPCMND statement in a message generation deck.

When you enter operator commands at the console, you can control parts of the simulation while WSim is running. When you enter operator commands on the OPCMND statement in a message generation deck, you can automate operation so that the network controls itself during operation.

The following sections describe entering operator commands at the console and with the OPCMND statement.

### Specifying operator commands at the console

WSim can use the MVS MODIFY command interface to receive the operator commands you enter at the operator console. These interfaces eliminate the need for WTOR processing by the main simulation task.

Although you can use the WTOR interface, the system MODIFY method is the preferred method for entering commands manually. This method also allows WSim to automatically echo the commands to the MVS job log. If you do not use the system MODIFY method, a time-stamped outstanding WTOR message is issued to enable you to enter console commands. In this case, the operator must preface each command with the number of the currently outstanding WTOR.

WSim uses the WTOR interface if WSim is started as a batch job.

### Specifying operator commands with the OPCMND statement

The OPCMND statement enables you to enter all operator commands other than console recovery subcommands from a message generation deck or STL program. Options are provided to include network names in the generated command. The commands are checked for validity by the console command routines when they are executed, not during network initialization. You should ensure that using these commands does not affect other user networks running at the same time.

Refer to *Creating WSim Scripts* and *WSim Script Guide and Reference* for more information about the OPCMND statement.

### Understanding the order of execution for operator commands

All operator commands, whether entered from the console or a message generation deck, are executed in the order they are entered. Each command is queued until the previously entered command has completed execution. The D (Dump), E (Restart Message Logging), O (Output), and T (Trace) commands that request printer or tape output are considered complete when the requests have been queued to the appropriate subtasks. Any other command is considered complete when all actions related to that command are accomplished.

## Specifying resources with operator commands

Operator commands allow you to modify and control WSim. To use them effectively, however, you should understand how operator commands affect simulated resources.

From the user's standpoint, WSim consists of networks with variable numbers of:

- CPI-C transaction programs (APPCLU)
- VTAM applications (VTAMAPPL)
- TCP/IP resources (TCPIP)

The level of qualification in the operator command determines the resource operated on by WSim. You specify the level of qualification either by entering the name of the desired resource or, in some cases, by entering the name of the desired resource preceded by all higher-level resource names required to identify the desired resource uniquely. For example, you can operate on the following resources:

- An entire network by specifying the network name
- A single LU by specifying the LU name.

You can operate on the networks independently of each other, so you can cancel one network without affecting any other networks. You can also change networks globally, with one command affecting all networks.

## Entering operator commands at the console

When you enter operator commands at the console, the command can start in any position. Blanks between commands and operands are ignored. The command can be in uppercase or lowercase letters. The maximum length of any command is 120 bytes, including blanks and comments.

Commands are presented to WSim by either the MODIFY or WTOR (Write to Operator with Reply) command interface when you enter the commands at the console. You can include comments with all commands containing at least one operand.

The following sections describe how to enter operator commands for MVS and TSO. See Chapter 20, "Specifying operator commands," on page 137 for a description of these commands.

### Entering operator commands on MVS with WSim as a started procedure

To operate WSim as a started procedure on MVS, use the following steps to enter operator commands:

- Start the simulation run with an MVS S (Start) command. For example:

  ```
  S WSIM.T,,,' execution parameters'
  ```

  This command starts WSim and assigns the name T as the identifier. From here on, T will be used when issuing the F (Modify) command for WSim.
- Enter the F (Modify) command using the following format:

  ```
  F T, command
  ```

  where *command* is the full operator command string to be executed. For example:

  ```
  F T,I NETNAME,L,S
  ```

When you use these steps to enter a command, WSim automatically echoes the command to the MVS job log. When you start WSim by using an MVS START command, WSim routes messages to specific operator consoles using the console identifier of the console where the START, STOP, or MODIFY commands are entered. WSim uses the full four-byte extended console identifier in routing responses and also preserves any command and response token (CART) that may have been supplied with the command. All messages written by WSim are flagged as responses so that they will appear as solicited messages when routed to a TSO extended console.

### Entering operator commands on MVS with WSim as a batch job

When operating WSim as a batch job on MVS, use the following steps to enter operator commands:

1. Submit the simulation run as an MVS batch job.
2. Wait for the following WTOR message to appear on your operator console:
   ```
   nn ITP001E  hh.mm.ss.th WSIM
   ```
3. Enter the operator command using the following format:
   ```
   R  nn, command
   ```
   where *nn* is the WTOR number and *command* is the full operator command string to be executed. For example:
   ```
   R 01,I NETNAME,L,S
   ```

### Entering operator commands when executing directly under TSO

To operate WSim under TSO, start the simulation run with a CLIST. You can invoke this CLIST on any command line, or you can use option 6 on the main ISPF menu.

You can enter operator commands from a TSO terminal. The commands do not have to be prefixed with the F (Modify) command. To initialize and start a network from a TSO terminal, enter:

```
I NETWORK1,L,S (Initialize, list, and start the network NETNAME)
```

You can stop WSim with the Z (Closedown) command:

```
ZEND
```

If your network writes messages to the screen so quickly that you cannot enter commands, press Enter. This halts the messages until you enter a command or press Enter again. However, messages generated during the time that output to the screen is halted are lost.

## Controlling WSim and simulated resources

This section discusses how to use the operator commands to control WSim and the simulated networks. See Chapter 20, “Specifying operator commands,” on page 137 for a complete discussion of the commands presented in this section.

### Starting and stopping WSim

You start WSim under MVS by submitting a job to the operating system, using a TSO CLIST, or starting it as a procedure. These start-up methods are discussed in “Running WSim” on page 66.

### Starting WSim

During start-up, WSim allocates buffers, initializes control blocks, and attaches sub-tasks. WSim then displays message ITP016I, the level and date of the release. WSim displays message ITP003I, WSIM INITIALIZATION COMPLETE, when start-up processing is complete. If you are using the MODIFY command interface, you may enter operator commands at this time. If you are using the WTOR interface, this message is followed by the WTOR message, ITP001E, which allows you to enter operator commands.

### Stopping WSim

You can stop WSim at any time by entering the ZEND operator command. When the ZEND command is processed, all active networks are stopped and canceled, WSim subtasks are detached, and control is returned to the operating system. When you are using the MODIFY interface, you may also stop WSim using the system operator STOP command (P *jobid*).

## Initializing and starting a network

After WSim start-up processing is complete, you must initialize a network to be simulated. The network may or may not have been preprocessed, but the network definition statements must reside in the INITDD data set, and the message generation deck statements must reside in the MSGDD data set.

### Initializing a network

The I (Initialize) command enables you to specify a member in the INITDD data set that is the network to be initialized. For example, the following command:

```
I TESTNET
```

causes WSim to initialize the network named TESTNET. The initialization consists of:

- Checking the syntax of all statements in the network
- Building all control blocks for the network

At the end of initialization, the network remains idle until you enter other operator commands to start the simulated resource activity.

The I command operands provide for listing the network statements on the printer during initialization and for automatically starting network activity at the end of initialization.

For example, the following command:

```
I TESTNET,LN,S
```

initializes the network TESTNET, prints the network definition statements, and starts activity for all simulated resources in the network. The LN operand is especially useful for obtaining a listing of the network definition statements without tying up the printer with long listings of the message generation decks.

Use the NTWRK and NTWRKL execution parameters to automate the network start-up process further. These parameters name a network to be initialized and started (NTWRK), and optionally listed (NTWRKL), automatically at the end of the WSim initialization processing.

### Starting network resources

If you do not use the automatic network start facilities, then you must use the S (Start) command to begin network activity. The S command can start activity for all initialized networks, for a single initialized network, or for a single TCPIP, APPCLU, or VTAMAPPL within a network. The following command:

```
S
```

starts activity for all simulated resources in all initialized networks.

You can enter the S command for specific TCPIP, APPCLU, and VTAMAPPL resources if you want to start network activity in a slower, more controlled manner. The FE (Future Event) statement and the STIME (Start Time) operand also contribute to the controlled starting of network resources. Refer to *Creating WSim Scripts* for discussions of these functions.

### Stopping network resources

You can stop the activity for all active networks, a single network or an APPCLU, VTAMAPPL, or TCPIP resource within a network by using the P (Stop Network) command. The P operator command with no operands stops all active networks. The operands name the network or resources you want to stop.

## Displaying the status of network resources

WSim provides two operator commands for displaying the status of currently active networks.

The G (Terminal Status Query) command displays those simulated terminals which are active, inactive, quiesced, ready, or terminated.

The Q (Query) command displays at the operator console the current status of a network resource (NETWORK, TCPIP, device, logical unit, APPCLU, transaction program, VTAMAPPL). The following sections explain the output provided for each type of resource.

### Query without specifying resources

If you enter a Q (Query) command without specifying resources, console message ITP006I or ITP012I is displayed for each initialized network. Refer to Figure 10 to see messages displayed when you specify the Q command and do not specify a resource, as shown below. An underscore indicates where a network name appears.

Command entered:

```
Q
```

*Figure 10. Query without resource specified*

```
ITP006I  NETWORK ________ STARTED
ITP012I  NETWORK ________ STOPPED
```

### Query network

You can enter a Q (Query) command and specify a network. Figure 11 shows the messages displayed when you specify the Q command and a network using the syntax shown below. Underscores indicate where resource-specific information appears. Refer to “Q-Query network resources” on page 156 for more information about specifying networks and network resources.

Command entered:

```
Q  ntwrk
```

```
ITP140I NTWRK=________ ITIME=___ SCAN=___,___,___ EMTRATE=_____,_____
ITP140I REPORT=____ CONRATE=___ MONCMND=___ DEBUG=___
ITP140I NSEQ __________
ITP140I NC01-05 __________ __________ __________ __________ __________
ITP140I NC06-10 __________ __________ __________ __________ __________
ITP140I NC11-15 __________ __________ __________ __________ __________
ITP140I NC16-20 __________ __________ __________ __________ __________
ITP140I NC21-25 __________ __________ __________ __________ __________
ITP140I NC26-30 __________ __________ __________ __________ __________
ITP140I NC31-35 __________ __________ __________ __________ __________
ITP140I NC36-40 __________ __________ __________ __________ __________
ITP140I NC41-45 __________ __________ __________ __________ __________
ITP140I NC46-50 __________ __________ __________ __________ __________
ITP140I NC51-55 __________ __________ __________ __________ __________
ITP140I NC56-60 __________ __________ __________ __________ __________
ITP140I NC61-63 __________ __________ __________
ITP140I NTWRK SWITCHES 01-04=____ 05-08=____ 09-12=____ 13-16=____
ITP140I 17-20=____ 21-24=____ 25-28=____ 29-32=____
ITP140I NTWRK UTI VALUE=_____
ITP140I INDIVIDUAL UTI VALUES:
ITP140I _______ =___ _____=___ _____=___ _____ =___.
ITP140I STARTED=___ HEADING=____________________
ITP140I SUBSTITUTE DECK=________ NCB LOCATION=______
ITP140I DEVs,LUs,TPs: ACTIVE=_____ INACTIVE=_____ QUIESCED=_____
ITP140I TERMINATED=_____ READY=_____
```

*Figure 11. Network query*

Detailed information about the data in Figure 11 is provided below.

**NTWRK=**
Is the symbolic network name from the NTWRK statement.

**ITIME=**
Is the time in minutes between network interval reports.

**SCAN=**
Gives the values of the variables x, y, and z defined under the SCAN operand of the NTWRK statement. For more information, refer to *WSim Script Guide and Reference*. If the scan function has not been defined, then "0,0,NONE" is displayed for *x, y,* and *z*.

**EMTRATE=**
Gives the expected message transfer rate followed by the current UTI adjustment interval in seconds.

**REPORT=**
Is either FULL, LINE, RATE, or NONE, indicating the type of interval report to be printed.

**CONRATE=**
Is either YES or NO, indicating whether or not the interval report message rates will be displayed at the operator console.

**MONCMND=**
Is either ON or OFF, indicating whether operator commands entered from a message generation deck with the OPCMND statement will be monitored at the operator console.

**DEBUG=**
Is either ON or OFF, indicating whether or not the network DEBUG option is active.

**NSEQ=**
Is the value of the network sequence counter.

**NC***n-n* The values of the network index counters are displayed with the headings NCn-n, where n can be from 01 to 4095. Only those counters defined will be displayed.

**NTWRK SWITCHES**
The 4095 network switches are displayed in groups of 4 bits on the lines beginning with NTWRK SWITCHES. Each bit has the value 1 (for ON) or 0 (for OFF).

**NTWRK UTI VALUE=**
Is the value of the network-level user time interval (UTI) in hundredths of seconds.

**INDIVIDUAL UTI VALUES:**
Is a list of individual UTIs that have been defined for the network. The label of the individual UTI is displayed, followed by the UTI value in hundredths of seconds. Refer to *Creating WSim Scripts* and *WSim Script Guide and Reference* for more information about defining and using individual UTIs.

**STARTED=**
Is either YES or NO, indicating whether or not the network has been started.

**HEADING=**
Is the 24-character heading for the interval report.

**SUBSTITUTE DECK=**
Is the name of the message generation deck to be substituted for a deck that causes automatic terminal recovery. If the SCAN operand has not been defined, then NONE is displayed.

**NCB LOCATION=**
Is the virtual storage address of the NCB control block. This field is displayed only if the network DEBUG option is active.

The ACTIVE, INACTIVE, QUIESCED, READY, and TERMINATED counts are based on started devices, logical units, and transaction programs only. The ACTIVE and INACTIVE counts represent the number of active and inactive devices, logical units, and transaction programs with respect to the values coded on the SCAN operand on the NTWRK statement. The QUIESCED count represents the number of quiesced devices, logical units, and transaction programs. The READY count represents the number of server transaction program instances that are waiting for incoming attach requests. The TERMINATED count represents the number of transaction program instances for which message generation activity has completed. Each device, logical unit, or transaction program started is included in one of the subject counts.

## Query TCP/IP connection, APPC LU, or VTAM application

You can enter a Q (Query) command and specify a TCP/IP connection, APPC LU or VTAM application. Figure 12 displays the messages when you specify the Q command and one of these resources. Underscores indicate where resource-specific information appears. Refer to “Q-Query network resources” on page 156 for more information about the different resources you can specify.

Command entered:

```
Q{ tcpip| appclu| vtamappl}
```

```
ITP141I NAME=________ TYPE=____ STARTED=___
ITP141I LSEQ __________
ITP141I LC01-05 __________ __________ __________ __________ __________
ITP141I LC06-10 __________ __________ __________ __________ __________
ITP141I LC11-15 __________ __________ __________ __________ __________
ITP141I LC16-20 __________ __________ __________ __________ __________
ITP141I LC21-25 __________ __________ __________ __________ __________
ITP141I LC26-30 __________ __________ __________ __________ __________
ITP141I LC31-35 __________ __________ __________ __________ __________
ITP141I LC36-40 __________ __________ __________ __________ __________
ITP141I LC41-45 __________ __________ __________ __________ __________
ITP141I LC46-50 __________ __________ __________ __________ __________
ITP141I LC51-55 __________ __________ __________ __________ __________
ITP141I LC56-60 __________ __________ __________ __________ __________
ITP141I LC61-63 __________ __________ __________
ITP141I LIN LOCATION=__________
```

*Figure 12. Query of a TCPIP, APPCLU or VTAMAPPL*

The statements in Figure 12 are displayed when you request a query for an APPC LU. When you request a query for a TCP/IP connection or VTAM application, these statements and the statements for a terminal are displayed. WSim maintains a terminal control block (TRM) associated with each TCP/IP connection or VTAM application. Thus, if a TCP/IP connection or VTAM application is being displayed, additional statements describing the terminal are displayed.

**NAME=**
Is the 8-character alphanumeric EBCDIC name. This field is only displayed for a TCP/IP connection, a CPI-C transaction program, or a VTAM application.

**TYPE=**
Is one of the following identifications: TCP/IP, APPCLU, or VTAMAPPL.

**STARTED=**
Is either YES or NO, indicating whether or not the resource has been started.

**LSEQ=**
Is the value of the line sequence counter.

**LC***n-n* The values of the line index counters are displayed with the headings LCn-n, where n is between 01 and 4095. Only defined counters are displayed.

**LIN LOCATION=**
Is the virtual storage address of the LIN control block. This field is displayed only if the network DEBUG option is active.

## Query device

You can enter a Q (Query) command and specify a device. Figure 13 displays the messages when you specify the Q command and a single device. Underscores indicate where resource-specific information appears. *num* may be the asterisk character (*) to indicate the last LU session number or last TP instance number. Refer to "Q-Query network resources" on page 156 for more information about the different resources you can specify.

Command entered:

```
Q { dev|  lu[- num]| appclu.tp[- num]| tp[- num]}
```

```
ITP143I NAME=________ __________ TYPE=________ QUIESCED=___
ITP143I TPTYPE=______ CPITRACE=_______ INSTANCE=_____________
ITP143I MSG DELAY=________________________ BLK DELAY=________________________
ITP143I WAIT=______ PRTSPD=_______ CURSOR=_______ INHIBIT=___
ITP143I SERVADDR=_______________ TCPSTATE=________ NXTSERVR=_______________
ITP143I WAIT EVENTS = ________
ITP143I ON EVENTS = ________
ITP143I INSERT PATH=____ PATHS=__ __ __ __ __ __ __ __ __ __ __ __
ITP143I CURRENT PATH=____ PATH ENTRY=___
ITP143I CURRENT DECK=________ CURRENT STATEMENT: WSIM=___, STL=___
ITP143I MSGTRACE=___ STLTRACE=___ INTERMESSAGE DELAY=_______
ITP143I ACTIVE UTI IS________=_____
ITP143I DSEQ __________
ITP143I DC01-05 __________ __________ __________ __________ __________
ITP143I DC06-10 __________ __________ __________ __________ __________
ITP143I DC11-15 __________ __________ __________ __________ __________
ITP143I DC16-20 __________ __________ __________ __________ __________
ITP143I DC21-25 __________ __________ __________ __________ __________
ITP143I DC26-30 __________ __________ __________ __________ __________
ITP143I DC31-35 __________ __________ __________ __________ __________
ITP143I DC36-40 __________ __________ __________ __________ __________
ITP143I DC41-45 __________ __________ __________ __________ __________
ITP143I DC46-50 __________ __________ __________ __________ __________
ITP143I DC51-55 __________ __________ __________ __________ __________
ITP143I DC56-60 __________ __________ __________ __________ __________
ITP143I DC61-63 __________ __________ __________
ITP143I DEV SWITCHES: 01-04=____ 05-08=____ 09-12=____ 13-16=____
ITP143I 17-20=____ 21-24=____ 25-28=____ 29-32=____
ITP143I TIME STAMP:________ DATA SENT:______________________
ITP143I IN HEX:_______________________________________
ITP143I TIME STAMP:________ DATA RECV:__________________
ITP143I IN HEX:_______________________________________
ITP143I DEV LOCATION=______
ITP143I HALF SESS=___ DFC STATE=_______
ITP143I LU-LU SESS=___ PARTNER=__________________
ITP143I SEND/RECV MODE=__________ __ HDX PROTOCOL=______ HDX STATE=____
ITP143I TH FORMAT=____ SEQ NO.: PRI-TO-SEC=_____ SEC-TO-PRI=_____
```

*Figure 13. Device query*

The response in Figure 13 is displayed when you request a query for any device capable of message generation.

**NAME=**
: Is the symbolic device name from the DEV, LU, or TP statement. The name is followed by:

| | |
|---|---|
| -NOT STARTED | if the line has not been started, |
| -INACTIVE | if the device is inactive, |
| -ACTIVE | if the device is active, |
| -READY | if the device is a CPI-C TP server instance that is waiting for an incoming attach request, |
| -TERMINATED | if the device is a CPI-C instance which has completed all message generation activity. |

**TYPE=**
: Is up to 8 characters describing the device type. These descriptions come from the valid values of the TYPE and LUTYPE operands on the DEV and LU statements. If this device is a CPI-C transaction program, the type reflects APPC-TP.

**QUIESCED=**
: Is either YES or NO, indicating whether or not the device is quiesced.

**MSG DELAY=**
: Gives the intermessage delay setting for the device. For information about the DELAY operand under the DEV, TP, or LU statement, refer to *WSim Script Guide and Reference*.

**BLK DELAY=**
: Gives the interblock delay setting for the device. NONE is displayed if this device has no interblock delays.

**WAIT=**
: Is either ON or OFF, indicating whether or not the device has been placed into a wait state during message generation.

**PRTSPD=**
: Is up to 5 digits specifying the processing delay in characters per second for received messages. This field is only displayed for nondisplay devices.

**CURSOR=**
: Gives the cursor position for a display device. The row number is displayed first, followed by the column number. If the cursor position has not been initialized, then NOT SET is displayed. This field is only displayed for display devices.

**INHIBIT=**
: Is either YES or NO, indicating whether or not the device's keyboard is inhibited from entering data. This field is only displayed for display devices.

**SERVADDR=**
: Specifies the IP dotted decimal address of the TCP/IP server to which you are currently connected.

**TCPSTATE=**
: Specifies the state of the device as it establishes a TCP/IP session and begins data exchange. Valid states are:

| | |
|---|---|
| CLOSED | The device has no current or pending connection. |
| OPENING | A connection attempt has been initiated for this device. |
| HOSTID | Local Host ID has been obtained. |
| SOCKET | The device obtained a socket for communication. |
| NEGO | The device is performing TCP/IP negotiations. |
| SOCKOPT | Socket options have been set. |
| READY | The TCP/IP connection for the device is ready and available for data transfer. |
| CLOSING | The device is closing its TCP/IP connection. |

**NXTSERVR=**
: Reflects the altered SERVADDR value if you altered the value. It is used upon reconnection, and becomes the current server at that point.

**WAIT EVENTS=**
: Gives a list of events upon which the terminal is waiting. All these events must be posted as complete before normal message generation for the terminal can resume.

**CPITRACE=**
: Indicates the level of CPI-C transaction programming tracing requested.

**INSTANCE=**
: Indicates the number of initial transaction program instances requested and the maximum number of instances that may be active concurrently.

**ON EVENTS=**
: Gives a list of events for which the terminal has issued an ON statement and have not been signaled. These events describe active “on” conditions.

**INSERT PATH=**
Is either ACT (an inserted path is active), PEND (an inserted path is pending active), or NONE (no inserted path).

**PATHS=**
Gives the names of the PATH statements defined for this device. As many PATH names as will fit are displayed on this display line, with additional path names displayed on the following lines without a label. NONE is displayed if PATH statements are not defined for the device.

**CURRENT PATH=**
Is the name of the PATH statement currently being processed. NONE is displayed if a PATH statement is not currently associated with the device.

**PATH ENTRY=**
Is the index into the current PATH statement for the message generation deck or STL procedure active for the device. This field is not displayed if CURRENT PATH=NONE.

**CURRENT DECK=**
Is the name of the message generation deck or STL procedure currently being processed. NONE is displayed if no message generation deck or STL procedure is active for the device.

**CURRENT STATEMENT: WSIM=**
Is the number of the message generation statement currently being processed.

**CURRENT STATEMENT: STL=**
Is the number of the STL statement currently being processed. If STLTRACE=NO is coded in your network definition, @program is not coded in your STL program, or the PROGRAM= execution parameter is not specified, STL statement numbers are not displayed.

**MSGTRACE=**
Is either YES or NO, indicating whether the message trace function is active.

**STLTRACE=**
Is either YES or NO, indicating whether the STL trace function is active.

**INTERMESSAGE DELAY=**
Is either ACTIVE or NOT ACTIVE, indicating the current intermessage delay status.

**ACTIVE UTI IS**
Indicates the individual UTI that is active for the device. It lists the label of the individual UTI and the UTI value in hundredths of seconds. (Refer to *Creating WSim Scripts* and *WSim Script Guide and Reference* for more information about defining and using individual UTIs.)

**DSEQ=**
Is the value of the device sequence counter.

**DC*n-n*** Gives the values of the device index counters, displayed with the headings DC*n-n*. n is between 01 and 4095. Only those counters defined are displayed.

**DEV SWITCHES**
The 4095 device switches are displayed in groups of 4 bits each on the lines beginning with DEV SWITCHES. Each bit has the value 1 (for ON) or 0 (for OFF).

**TIME STAMP**
Is the time when the last message was transmitted or received by the device.

**DATA SENT**
Gives the first 20 characters of the message in EBCDIC. Unprintable characters are displayed as periods. If more than 20 bytes of data were sent and you coded the CRDATALN operand for this device with a value greater than 20, the remaining data is displayed in 32-byte segments after the hexadecimal translation of the first 20 bytes. If the device has not transmitted a message, then the entire display line beginning with TIME STAMP: is replaced by NO MESSAGE TRANSMITTED. If the device is a CPI-C transaction program and no conversations are currently active, the DATA SENT and DATA RECEIVED lines are replaced with NO ACTIVE CONVERSATIONS.

**IN HEX**
Gives the hexadecimal representations of the portion of the transmitted or received message on the previous line.

**DATA RECV**
Gives the first 20 characters of the message in EBCDIC. Unprintable characters are displayed as periods. If more than 20 bytes of data were received and you coded the CRDATALN operand for this device with a value greater than 20, the remaining data is displayed in 32-byte segments after the hexadecimal translation of the first 20 bytes. If the device has not received a message, then the entire display line beginning with TIME STAMP: is replaced by NO MESSAGE TRANSMITTED. If the device is a CPI-C transaction program and no conversations are currently active, the DATA SENT and DATA RECEIVED lines are replaced with NO ACTIVE CONVERSATIONS.

**DEV LOCATION=**
Is the virtual storage address of the DEV control block. This field is displayed only if the network DEBUG option is active.

The next five display lines are only displayed for SNA devices.

**HALF SESS=**
Is either PRI or SEC, indicating whether this logical unit comprises a primary or secondary half session.

**DFC STATE=**
Is one of the following data flow control states: ACTIVE (the LU has received an Activate Logical Unit command), or RESET (the LU has not received an Activate Logical Unit command or has received a Deactivate Logical Unit command).

**LU-LU SESS=**
Is either YES or NO, indicating whether or not the logical unit is currently part of an LU to LU session.

**PARTNER=**
Is the name of the partner LU with which this LU half-session is in session. This field is displayed only if LU-LU SESS=YES.

The next two display lines are displayed only if LU-LU SESS=YES and a BIND command has been successfully processed.

**SEND/RECV MODE=**
Is one of the following: FDX (full duplex), HDX,CONTENTION (half duplex, contention), or HDX,FLIP-FLOP (half duplex, flip-flop).

**HDX PROTOCOL=**
Is either WINNER or LOSER if the send/receive mode is half duplex, contention.

**HDX PROTOCOL=**
Is either SPEAKER, BIDDER, or NO BRKT (no brackets used) if the send/receive mode is half duplex, flip-flop.

**HDX STATE=**
Is either SEND (device can send a message), RECV (device can receive a message), or CONT (device is in contention situation).

**TH FORMAT=**
Is either FID1, FID2, or FID3.

**PRI-TO-SEC=**
Is the primary-LU-to-secondary-LU sequence number.

**SEC-TO-PRI=**
Is the secondary-LU-to-primary-LU sequence number.

### Query save and user areas

You can enter a Q (Query) command and specify a save area or a user area. Figure 14 shows the messages when you specify the Q command and a save area or a user area.

Command entered:

```
Q NET1,N1+,64
```

```
ITP219I NET1 NETWORK SAVE AREA 1, TOTAL LENGTH 63
ITP220I +0 E3C8C9E2 40CAE240 E3C8C540 E2E3C1D9 THIS IS THE STAR
ITP220I +16 E340D6C6 40E3C8C5 40D5C5E3 E6D6D9D2 T OF THE NETWORK
ITP220I +32 E2C1E5C5 40C1D9C5 C140C6D6 D940E3C8 SAVE AREA FOR TH
ITP220I +48 C540D9C5 E2D6E4D9 C3C540D5 C5E3F1 E RESOURCE NET1
```

*Figure 14. Save area or user area query*

The above example shows a Query request for the first 64 bytes of network N1 save area. The phrase "THIS IS THE START OF THE NETWORK SAVE AREA FOR THE RESOURCE NET1" was previously saved in NET1's first network save area. The first portion of each message shows the positive decimal offset into the specified area. The remaining portion of each message shows up to 16 bytes of requested save area or user area data, followed by its EBCDIC representation.

**Note:** The length of save areas is determined by the data saved into them; the length of user areas is specified in the network definition.

## Using service facilities

The service facilities enable you to obtain detailed information about how WSim is running and about communications with the system under test. The facilities include a trace of the dispatcher activity and dumps of control blocks.

### Dispatcher trace

The dispatcher trace is a table containing 32-byte entries that describe the sequence of activity within WSim. An entry is made in the table every time an element of work is dispatched to a processing routine, a module obtains a buffer, or a module frees a buffer.

The dispatcher trace is always active, using a table size of 60K bytes. You can dump the dispatcher trace table to the printer by entering the T (Trace) operator command:

```
T DSP
```

### Dumping control blocks

To solve certain problems, you may need a dump of the control blocks associated with a particular simulated network. The D (Dump) command enables you to specify the subset of control blocks dumped for a network. For example, the command:

```
D TESTNET,N
```

dumps the network control block to the printer.

## Controlling resources on a network

WSim provides several operator commands to enable you to control and change the simulated networks dynamically.

### Starting and stopping resources

You can use the S (Start Network Resources) command and the P (Stop Network Resources) command to start and stop activity on the TCPIPs, APPCLUs, and VTAMAPPLs in any active networks. You can use these commands globally, such as to stop all of the TCPIPs, APPCLUs, and VTAMAPPLs in a single network (for example, P TESTNET). By using these commands on specific resources, such as in stopping a single TCPIP, APPCLU and VTAMAPPL (for example, P010020), you can control the number of active simulated devices.

The different ways that you can perform SNA session initiation by WSim are discussed in *Creating WSim Scripts*.

### Resetting a network

After you initialize a network and start the simulation, you can use the R (Reset) operator command to reset the network to its initial status if you have already stopped the entire network with a P (Stop Network) operator command. The R operator command operates on an entire network or all networks but will not operate on a single resource within a network.

When WSim resets a network, the following functions are performed:

- All sequence and index counters are cleared, except for device sequence counters, which are reset to their initial values.
- The message generation path for a device is reset to the first path specified for the device.
- The FRSTTXT deck for a device is marked as the first message generation deck to be executed.
- All interval report statistics are cleared.

- All SNA devices are marked as not in session.
- All random number generator seeds are reset to their initial values.
- All RSTATS are cleared.

The R command does not reset network parameters that have been changed by an A (Alter) command, such as user time interval (UTI) or intermessage delay values.

## Changing network parameters

The A (Alter) command enables you to alter the values of network parameters dynamically. You can use this command to change a parameter for all initialized networks, for a single initialized network, or for a specific resource in a network. The following examples show some of the formats of the A command:

**A TESTNET,U=10**
Sets the network-level UTI to 10 for the network TESTNET.

**A I=5**
Sets the interval report time to 5 minutes for all networks.

**A 010040.TERM1.DEV2,M=A20**
Sets the intermessage delay for device DEV2.

**A DEV2,PRTSPD=120**
Sets the print speed for device DEV2.

**A DEV1,5=ABCDEFGHI**
Sets the contents of DEV1's save area 5 to the EBCDIC string 'ABCDEFGHI'.

**A LUB,PATH=6**
Sets message generation for LUB to execute PATH 6 once and then return to normal path selection.

**A TESTNET,PATH=(7)**
Sets message generation for all devices in network TESTNET to execute only PATH 7.

**A NSEQ=0**
Sets the network sequence counter to 0 for all networks.

## Canceling network resources

You can use the C (Cancel) operand command to purge network resources from WSim without affecting other networks. The C command cancels the execution of all networks, a single network, or a network resource, then removes the definitions of the canceled resources from WSim. The entity to be operated on by the C command can be active or inactive. You must reinitialize the network to reuse a network or network resource that has been canceled.

## Using console recovery

The console recovery feature of WSim allows you to interrupt message generation processing for a specified device from the operator's console. The device may be active or inactive when console recovery is entered. The conditions under which a device is classified as inactive are discussed in *WSim User Exits* and *Creating WSim Scripts*. You can use the F (Console Recovery) command to enter console recovery mode for a device. For example, the command:

```
F DEV231
```

invokes console recovery for the device named DEV231. While console recovery is active, WSim provides the WTOR message ITP098E to let you specify subcommands to generate data for the device.

When you enter console recovery mode, the LOGICAL WAIT indicator is reset, delays are canceled, event wait conditions are reset, and the INPUT INHIBITED indicator is reset for a 3270 device. If either of these conditions is the reason that the device is inactive, then entering and exiting console recovery mode reactivates the device. The state of a simulated terminal is not automatically reset, however.

## Using online response-time statistics

You can use the online Response-Time Statistics feature, RSTATS, to provide online response-time calculations for terminals simulated by WSim. RSTATS measures the time it takes to enter a command at the terminal and receive a response from the system under test.

RSTATS only collects response-time statistics for those simulated resources that generate messages. These include devices and LUs.

### RSTATS output

WSim supports RSTATS for all terminal types. When you activate the RSTATS option in the network definition, you can obtain the statistics shown in Figure 15 by entering the W (RSTATS) operator command for a particular device:

```
RESPONSE TIME STATISTICS FOR DEVICE ABC
AT 11.32.45.21
PROCESS SYSTEM PROCESS ACTUAL
AVERAGE 1.23 1.34
MOST RECENT 1.56 1.89
LOW 1.00 1.00
HIGH 2.00 2.00
TOTAL RESPONSES 100 100
```

*Figure 15. Example statistics provided by RSTATS*

The statistical measures are defined as follows:

| | |
|---|---|
| **Average** | This measure gives the running average of all response times since the statistics were started. This is the statistical mean. |
| **Most Recent** | This measure is the last response-time value calculated for this device before you entered the W operator command. |
| **Low** | This measure is the lowest response time since the statistics were started. |
| **High** | This measure is the highest response time since the statistics were started. |
| **Total Responses** | This measure is the total number of responses or completed transactions for this device since the statistics were started. |

**Note:** All response times are given in seconds. If no responses have been processed, all values will be zero.

You can use two methods to report these statistics: Process System and Process Actual. Process System measures the time it takes the system under test to return a response after the command is read. Process Actual measures the response time that you can see, from the time you press the Enter key until the response is written on the panel. Refer to *WSim Utilities Guide* for a complete description of the Response Time Utility that defines Process System and Process Actual in greater detail.

In most cases, the results from RSTATS should closely resemble the results from the Response Time Utility because RSTATS processes the same LOG control block immediately after it is written to the log data set. Like the Response Time Utility, RSTAT processes every transmit and receive record. However, when you use RSTATS, you cannot define logical transaction processing (BTRANS and ETRANS). In addition, certain messages are discarded for 3270 and LU7 devices.

Because RSTATS calculates response times during operation, some of the more complex processing available when you use the Response Time Utility is unavailable.

### The RSTATS operand

To enable the response-time feature for a particular device, code the following in the network definition:

```
RSTATS=YES
```

You can code RSTATS=YES on DEV or LU statements. You can also code it on higher-level statements and allow it to default to lower-level statements.

RSTATS=YES specifies that response statistics will be accumulated and reported when you enter the W operator command for the device.

RSTATS=NO specifies that no response statistics will be kept for this device.

If you specify RSTATS=NO for a device in the network definition, you cannot activate RSTATS for that device at any time during the simulation. This is due to the storage allocation necessary at initialization time.

### Resetting response statistics

You can use the following A (Alter) operator command at any point during operation to reset the response-time statistics:

```
A { name},RSTATS
```

where *name* is any specification operand for the command.

This command causes all fields in the response-time control blocks to be instantly set to zero, which is the initial state before the network is started. This prevents previous statistics from biasing future figures and can be especially useful when you adjust UTI values or dynamically change test system parameters during the run.

Response-time statistics are also reset when you enter the R (Reset) operator command.

## Routing messages

The following sections discuss how you can route messages to the console and to the log data set.

## Routing messages to consoles

Two kinds of messages appear at the console during a WSim run: write-to-operator (WTO) messages that come directly from WSim and system messages. The routing of WTO messages depends on which command interface, MODIFY or WTOR, is in use and whether you specified the ROUTCDE execution parameter (MVS only).

See *WSim Messages and Codes* for details about system messages and codes.

### Routing console messages with MODIFY

One advantage of the MODIFY command interface is that WSim routes responses to operator commands only to the console from which you entered the request using the full four-byte extended console ID and preserving the command and response token (CART), if any, supplied with the command. WSim routes responses to operator commands entered with the OPCMND statement to the console from which you initialized the network.

**Note:** If you are operating WSim under MVS and you specify the ROUTCDE execution parameter, you override the MODIFY routing mechanism. In this case, WSim routes messages as if you were not using the MODIFY interface.

### Routing console messages without MODIFY

For systems with multiple console support (MCS), WSim routes all messages to the consoles that specify the proper routing code. The default ROUTCDE value is 8, but you can change the value with the ROUTCDE execution parameter. For systems without MCS, the master console receives the messages. All console messages have description codes that indicate whether or not action is required on the message. In addition, all messages contain message numbers so you can find out the meaning of the message. For a description of these messages, refer to *WSim Messages and Codes*.

## Routing messages to log data sets

WSim writes all messages normally written to the operator console, other than message numbers ITP001E and ITP0301, to the appropriate log data set, assuming the log data set is ready. If you are using the NTWRKLOG data set for a network, WSim logs console messages associated with that network to that data set. Otherwise, WSim logs all console messages to the general log data set.

WSim associates the following messages with a particular network and logs them to the NTWRKLOG data set:

- CNSL messages resulting from message generation script WTO
- Comments about the state of a network.

WSim can also generate messages as a result of commands issued using the OPCMND statement in a message generation deck. WSim logs these messages to the general log data set. You can monitor these messages at the operator console.

# Chapter 16. Using operator reports

This chapter describes the types of operator reports WSim generates:

- Interval Report
- End of Run Report
- Inactivity Reports.

## Using interval reports

The interval reports provided by WSim report the activity and status of networks and network resources. Each interval report prints at the interval time you specify with the ITIME operand on the NTWRK statement. For each report, WSim accumulates network and resource statistics until you cancel the network or WSim completes an R (Reset) operator command.

The Interval Report is an online printout of network activity and status that provides information about each resource in the network. However, you may specify condensed forms of the report by using the REPORT operand on the NTWRK statement or the X operand on the A (Alter) operator command.

The Interval Report includes the following information for each resource in the network:

- Network name
- Network header
- Network-level user time interval value (NTWRK UTI)
- APPCLU and TP names
- VTAMAPPL and LU names
- TCP/IP connection and DEV names
- Current® resource status:

  **A** Active

  **C** Canceled

  **I** Inactive

  **P** Stopped

  **Q** Quiesced

  **R** Ready

  **S** Started

  **T** Terminated

  **W** Wait indicator on.

  Status for an LU in a simulated subarea is not printed until the LU is in an SNA session.
- Number of messages (complete transactions) sent from each terminal or device (maximum of 99,999 displayed). For CPI-C transaction programs, this is the total number of send verbs issued for which data is sent to the partner application, plus the number of send error verbs issued for which log data is sent to the partner application, plus the number of attach requests sent using the allocate verb.

- Number of messages (complete transactions) received by each terminal or device displayed (maximum of 99,999 displayed). For CPI-C transaction programs, this is the total number of receive verbs issued for which data is received from the partner application, plus the number of attach requests received using the accept verb (at most one per TP instance).
- Number of SNA responses sent and received (maximum of 99,999 displayed):
  - CPI-C simulation not applicable.
  - VTAMAPPL LUs: the number of definite and exception responses sent and received.
- Totals of all statistics for each APPCLU
- Totals of all statistics for each VTAMAPPL
- Totals of all statistics for each TCP/IP connection
- Totals of all statistics for the entire network
- Totals of all statistics for the entire network for the last reporting interval only (not cumulative)
- Rates per minute of each statistic computed for the last reporting interval only (not cumulative).

**Note:** Excessive monitor output may interfere with rate-per-minute calculations on the Interval Report.

Figure 16 shows an example of the Interval Report.

```
                                            INTERVAL REPORT
       NETWORK WSIMNET1                                    Sample WSim Network                             NTWRKUTI 100
                        STATUS    MESSAGES                       RESPONSES
                             RECEIVED   SENT            RECEIVED         SENT
                                                        DEF     EXC    DEF     EXC
APPCLU  APPCLUS           S
  TP      TPSERVER-1      T       2        1             0        0       0       0
  TP TOTALS (1:5)                10                      0                0
                                           5                      0               0
  APPCLU TOTALS                  10                      0                0
                                           5                      0               0
APPCLU  APPCLUC           S
  TP      TPCLIENT-1      A W     5        10            0        0       0       0
  APPCLU TOTALS                   5                      0                0
                                           10                     0               0
VTAMAPPL VA1              S
  LU       LULU2-1        A W     9        7             1        0       3       0
  VTAMAPPL TOTALS                 9                      1                3
                                           7                      0               0
TCP/IP  TCPIP1            S
    DEV    DEVTN32E       A       11       7             0        0       0       0
    DEV    DEVSTCP        A W     5        5             0        0       0       0
  TCP/IP TOTALS                   16                     0                0
                                           12                     0               0
  CUMULATIVE TOTALS               40                     1                3
                                           34                     0               0
  INTERVAL TOTALS                 16                     0                0
                                           12                     0               0
RATE (PER MINUTE)                 16                     0                0
                                           12                     0               0
```

*Figure 16. Interval report*

**Note:** For CPI-C transaction programs with multiple instances, a total line reports the totals for all instances of the transaction program. The range of instances appears parenthetically on the totals line. For example, the (1:3) on the TP totals line indicates that the totals represent TP instance 1 through TP instance 3. If you code TPSTATS=NO, or take the default, WSim does not maintain individual statistics for terminated TP instances. In this case, only active instances have an individual statistics line on the interval report. However, the TP totals line reflects totals for all simulated instances.

## Using end of run reports

The End of Run Report is an interval report that provides you with summary data for the simulated network. It prints automatically at the end of each run when the network is canceled. The format is the same as that for the Interval Report. Figure 17 shows an example of the End of Run Report.

```
                                                       END OF RUN REPORT
               NETWORK WSIMNET1                        Sample WSim Network                          NTWRKUTI 100
                               STATUS MESSAGES                     RESPONSES
                                    RECEIVED SENT          RECEIVED         SENT
                                                          DEF     EXC    DEF    EXC
APPCLU APPCLUS
  TP  TPSERVER-1                         2     1           0       0      0      0
  TP TOTALS (1:5)                       10                 0              0
                                               5                   0             0
  APPCLU TOTALS                         10                 0              0
                                               5                   0             0
APPCLU APPCLUC
  TP TPCLIENT-1                          5     10          0       0      0      0
  APPCLU TOTALS                          5                 0              0
                                               10                  0             0
VTAMAPPL VA1
  LU LULU2-1                             9      7          1       0      3      0
  VTAMAPPL TOTALS                        9                 1              3
                                                7                  0             0
TCP/IP TCPIP1
    DEV   DEVTN32E                      13      7          0       0      0      0
    DEV   DEVSTCP                        5      5          0       0      0      0
  TCP/IP TOTALS                         18                 0              0
                                               12                  0             0
  CUMULATIVE TOTALS                     42                 1              3
                                               34                  0             0
```

*Figure 17. End of Run Report*

**Note:** For CPI-C transaction programs with multiple instances, a total line reports the totals for all instances of the transaction program. The range of instances appears parenthetically on the totals line. For example, the (1:3) on the TP totals line indicates that the totals represent TP instance 1 through TP instance 3. If you code TPSTATS=NO, or take the default, WSim does not maintain individual statistics for terminated TP instances. In this case, only active instances have an individual statistics line on the interval report. However, the TP totals line reflects totals for all simulated instances.

## Using inactivity reports

The inactivity reports contain information about the status of terminals and devices in the network. To receive these reports, you must code the SCAN operand on the NTWRK statement or use the S operand of the A (Alter) command. These operands must specify a time interval between reports that is greater than zero. The criteria for an inactive terminal are discussed in *Creating WSim Scripts*. Note that simulated CPI-C TP instances are never considered inactive in this sense and therefore never appear on inactivity reports.

For each inactive resource in the network, the Inactivity Report contains the following information:

- Whether the device was last transmitting or receiving (T/R=)
- Message generation deck name, if any (ACTIVE DECK=)
- Time of the last message transmitted
- Time of the last message received
- Last message transmitted
- Last message received.

All or parts of the last messages transmitted and received by the terminal are printed in EBCDIC and hexadecimal notation. The maximum amount of data printed is determined by the value of the CRDATALN operand for this terminal. If the CRDATALN operand is not coded, 20 bytes of data are printed.

The Inactivity Report has a time stamp so that you can determine the length of time a terminal has been inactive. If no terminals or devices are inactive, the report contains the network name and the title INACTIVITY REPORT followed by the lines ALL TERMINALS ACTIVE and END OF REPORT. Figure 18 shows an example of the Inactivity Report.

```
         NETWORK  SAMPNET                           INACTIVITY REPORT
            VTAMAPPL=APPL1        LU=PLU1-1         T/R=T ACTIVE DECK=PLUDECK
           LAST TRANSMITTED     TIME  9.08.43.60
 00000000   F1C3                                                                        *1C                              *
           LAST RECEIVED        TIME  9.08.43.10
 00000000   7DC4D811 407DD4C5 E2E2C1C7 C540D5E4   D4C2C5D9 40F0F0F5 40C6D9D6 D440C4C5   *'DQ. 'MESSAGE NUMBER 005 FROM DE*
 00000020   E5F1C140 C1C2C3C4 C5C6C7C8 C9D1                                             *V1A ABCDEFGHIJ                  *
------------------------------------------------------------------------------------------------------------------------------------
            VTAMAPPL=APPL1        LU=PLU1-2         T/R=T ACTIVE DECK=PLUDECK
           LAST TRANSMITTED     TIME  9.08.43.62
 00000000   F1C3                                                                        *1C                              *
           LAST RECEIVED        TIME  9.08.43.11
 00000000   7DC4D811 407DD4C5 E2E2C1C7 C540D5E4   D4C2C5D9 40F0F0F5 40C6D9D6 D440C4C5   *'DQ. 'MESSAGE NUMBER 005 FROM DE*
 00000020   E5F1C240 C1C2C3C4 C5C6C7C8 C9D1                                             *V1B ABCDEFGHIJ                  *
------------------------------------------------------------------------------------------------------------------------------------
              TCPIP=TCONN1     DEVICE=DEV11         T/R=R ACTIVE DECK=WAITREDY
           LAST TRANSMITTED     TIME  9.08.44.01
 00000000   7D5BF011 D840C5D9 C9C3C811 D950C6F0   F0E3C2C1                              *'$0.Q ERICH.R&F00TBA            *
           LAST RECEIVED        TIME  9.08.45.99
 00000000   F1C2115B 5F1DC111 5D6B1D60 C8D6D3C4   C9D5C740                              *1B.$-.A.),.-HOLDING             *
------------------------------------------------------------------------------------------------------------------------------------
LINE=010033 TERMINAL=TERM1     DEVICE=DEV1A         T/R=T ACTIVE DECK=SLUDECK
           LAST TRANSMITTED     TIME 9.08.46.37
 00000000   7DC4D811 407DD4C5 E2E2C1C7 C540D5E4   D4C2C5D9 40F0F0F5 40C6D9D6 D440C4C5   *'DQ. 'MESSAGE NUMBER 005 FROM DE*
 00000020   E5F1C140 C1C2C3C4 C5C6C7C8 C9D1                                             *V1A ABCDEFGHIJ                  *
           LAST RECEIVED        TIME 9.08.46.35
 00000000   F1C3                                                                        *1C                              *
------------------------------------------------------------------------------------------------------------------------------------
LINE=010033 TERMINAL=TERM1     DEVICE=DEV1B         T/R=T ACTIVE DECK=SLUDECK
           LAST TRANSMITTED     TIME 9.08.46.58
 00000000   7DC4D811 407DD4C5 E2E2C1C7 C540D5E4   D4C2C5D9 40F0F0F5 40C6D9D6 D440C4C5   *'DQ. 'MESSAGE NUMBER 005 FROM DE*
 00000020   E5F1C240 C1C2C3C4 C5C6C7C8 C9D1                                             *V1B ABCDEFGHIJ                  *
           LAST RECEIVED        TIME 9.08.46.56
 00000000   F1C3                                                                        *1C                              *
------------------------------------------------------------------------------------------------------------------------------------
 END OF REPORT
```

*Figure 18. Inactivity Report*

# Chapter 17. Controlling message logging

The message logging facility is an important tool to use for debugging message generation decks and network definition statements. This chapter describes how to control message logging using operator commands. For information about controlling message logging using network definition statements, refer to *Creating WSim Scripts*. For information about understanding and formatting the log data set, refer to *WSim Utilities Guide*.

## What is message logging?

The message logging facility, when active, writes messages to the log data set containing all data that simulated resources transmit or receive in a specified network. Most 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 or FILEDEF 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 for the entire network by specifying MLOG=NO on the NTWRK statement, or you can override the network definition for a single TCP/IP connection by using the MLOG operand on the TCPIP statement. You can also use both the NTWRK and TCPIP statements.

Furthermore, you can code the NTWRKLOG statement in the network definition to specify the log data set for this network. In this way, you can run multiple networks simultaneously and collect log data for each network in a separate data set.

By inspecting the printed records in the log data set after a simulation run, you can gain information about the behavior of your teleprocessing network. For example, you can determine whether or not new application programs operated correctly. You can also obtain an estimate of your system's performance by using the time stamps in each record to compute the response times between the messages transmitted and received by the simulated terminals.

Refer to *WSim Utilities Guide* for a discussion of message logging, time stamps, and the utility programs that provide offline analysis of the log data set.

## Logging messages

This section discusses the facilities that control message logging in MVS. Because WSim also allows you to specify separate logging machines for each network (network log data set), this section also contains a discussion of the NTWRKLOG data set. If you have defined the message logging facility for your network, you can use LOGDD DD statement parameters in the JCL you use to run WSim to control the writing of messages on a tape or in a DASD data set.

For tapes, use the VOLUME parameter to specify that WSim use multiple tape volumes for the log data set. If you specify multiple volumes in this way, WSim automatically requests that a new tape volume be mounted when the previous volume is full. For example, the following DD statement specifies that a maximum of four tape volumes be used for the log data set:

```
//LOGDD DD DSN=WSIM.MSGLOG,DISP=(NEW,KEEP),UNIT=TAPE,
//         VOLUME=(,,,4,SER=(00001,00002,00003,00004)),
//         LABEL=(,BLP)
```

To change tapes, use the E (Restart Message Logging) operator command to cause WSim to force an end-of-volume condition on the current log tape volume. After unloading the tape, you can mount the next volume according to the requirements of your operating system. If you do not specify multiple volumes on the LOGDD statement, the tape rewinds and message logging resumes. Previous data will be lost.

The following DD statement defines the log data set for a DASD:

```
//LOGDD DD DSN=WSIM.MSGLOG,DISP=(NEW,KEEP),UNIT=3380,
// VOL=SER=YOURPK,SPACE=(CYL,(1,1))
```

When logging to disk on MVS, the E command causes an error condition that results in logging being stopped, but logging is immediately restarted. Message logging continues as usual, but messages logged prior to the E command are lost unless you give the log data set a DISP=MOD indication in the JCL.

For more information about restarting message logging, see “Restarting message logging” on page 123.

WSim maintains, by default, five buffers of 8192 bytes each for writing messages to the log tape or data set. For networks that achieve high message transfer rates or operate with unusually large messages, the default buffers may not be sufficient to contain all the data to be logged while waiting on an I/O operation to the log data set. In this situation, the system loses data and writes message ITP030I to the operator console. To increase the size and number of message logging buffers, you can specify the BLKSIZE and NCP subparameters of the DCB parameter. Use BLKSIZE to specify the maximum length of each buffer in bytes. Use NCP (Number of Channel Programs) to specify the number of buffers.

In the following example, the DD statement specifies that WSim allocate five buffers of 23476 bytes each for logging messages:

```
//LOGDD DD DSN=WSIM.MSGLOG,DISP=(NEW,KEEP),UNIT=TAPE,
//         LABEL=(,BLP),VOLUME=SER=00001,
//         DCB=(BLKSIZE=23476,NCP=5)
```

**Note:** For specific details about coding the VOLUME and DCB DD statement parameters, refer to the appropriate JCL reference manual for your operating system.

You can also specify NCP as an execution parameter; however, any LOGDD DD statement NCP value overrides the execution parameter value.

## Inhibiting the logging of console messages

Normally, WSim writes all console messages (ITP001E - ITP399I) and all log data messages (ITP400I - ITP499I) to a log data set. The Loglist Utility identifies console messages as CNSL records; log data messages are identified as INFO, STRC, MTRC or CTRC records. Refer to *WSim Utilities Guide* for information about running the Loglist Utility.

You may want to inhibit the display and logging of some of these messages. To inhibit these messages, use the INHBTMSG operand on the NTWRK statement, the MSGn= operand of the A (Alter) operator command, or both.

Because the INHBTMSG operand must be coded within the network definition, you must know which message or messages you want to inhibit prior to the WSim run. To inhibit a message during the run, you can use the A (Alter) operator command.

For more information about defining the configuration of the simulated network and inhibiting messages with the INHBTMSG operand, refer to *Creating WSim Scripts* and the *WSim Script Guide and Reference*.

The following example shows how you can use the A (Alter) operator command to inhibit the display and logging of message ITP137I for network NETA:

```
A NETA,MSG137=OFF
```

You can use the A (Alter) operator command to change the status of an inhibited message to one that is not inhibited so that subsequent issuances of the message will be displayed and logged.

In the following example, the A (Alter) operator command causes WSim to display and log message ITP137I until the end of the simulation run or until you inhibit it again using the A (Alter) operator command.

```
A NETA,MSG137=ON
```

For more information about inhibiting messages using the A (Alter) operator command, refer to “Using operator commands” on page 138.

## Restarting message logging

The E (Restart Message Logging) operator command gives you control over the message logging and network logging facilities. You can specify a network name operand with the E operator command to restart message logging for the associated NTWRKLOG data set. If you do not specify a network name, WSim restarts message logging for the general log data set. For more information about coding the *ntwrk* operand on the E operator command, refer to “Using operator commands” on page 138.

When you log messages on a tape on MVS and specify on the LOGDD DD statement that the log tape has multiple volumes, you can use the E operator command to force an end-of-volume condition on the currently mounted tape while message logging is active. The tape will be unloaded and you can then mount the next volume.

When logging to disk on MVS, the E command causes an error condition that results in logging being stopped, but logging is immediately restarted. Message logging continues as usual, but messages logged prior to the E command are lost unless you give the log data set a DISP=MOD indication in the JCL.

When an I/O error occurs on the log data set, WSim suspends message logging and notifies the operator with message ITP048I. You can use the E command to attempt a restart of the message logging facility by reopening the DCB associated with the log data set.

# Chapter 18. Using the Display Monitor Facility

The Display Monitor Facility allows you to see screen images or data streams from simulated devices on a real 3270 display. It is a VTAM application program that is part of WSim.

You can use the Display Monitor Facility in two different ways: to see the screen images of a simulated device (3270 only) or to watch the data streams that a simulated device sends and receives (all devices). Both of these help you develop and debug tests for your simulated devices. In addition, you can use the Display Monitor Facility to show interactions with host applications for user education or product demonstrations.

VTAM Version 3 Release 1 and any subsequent releases of VTAM support the Display Monitor Facility. It runs under MVS. The facility is activated when you start WSim with the DMAPPL execution parameter specified. Refer to "Using WSim execution parameters" on page 95 for a description of the WSim execution parameters. The DMAPPL parameter value specifies the name of the APPL defined to VTAM to be used by the Display Monitor Facility. The following example shows how to code the APPL in VTAMLST:

```
DISPMON APPL AUTH=ACQ
```

You may start monitoring simulated devices by either:

- Entering the M (Monitor) operator command
- Logging onto the Display Monitor Facility (DMAPPL value) and responding to a selection panel.

For information on how to define an APPL resource to VTAM, refer to *VTAM Network Implementation Guide*.

## Using the Display Monitor Facility

You may go through several stages of debugging when you code the network definition statements and create the message generation decks. The Display Monitor Facility helps speed up this process by helping you debug your scripts.

### Debugging scripts

Even if you use one of the script generating utilities (refer to *WSim Utilities Guide* for more information), you may need to modify your message generation decks. The Display Monitor Facility monitors a device as it goes through a script and helps show errors at run time that otherwise you could detect only if you ran the Loglist Utility after completing the simulation. The Display Monitor Facility can help you detect the following types of errors:

**Invalid cursor position**

By inserting the MONITOR statement into problem areas of your message generation decks and viewing the screen images during a run, you can detect problems related to cursor position. The cursor may be positioned in a protected field or the wrong input field. Using the MONITOR statement lets you keep the screen active on the monitoring display so that you can investigate the problem.

**Format of the screen image**

A monitoring display shows you what your panel looks like at any given time. Using LOGDSPLY and running the Loglist Utility has the same result, but it also requires you to stop WSim and run the Loglist Utility. With the Display Monitor Facility, you can detect this error during run time, correct it, and run WSim again, eliminating a step when you debug your script.

**Hung devices**

When you use the Display Monitor Facility with VIEW=SCREEN, UPDATE=XMITRECV, and SOURCE=BLOCKS specified, the monitoring display shows the last screen image transmitted or received. This may help you to determine why the device is no longer sending data.

**Data flow errors**

By using the Display Monitor Facility with VIEW=DATA specified, you can view the transmitted and received data streams for the simulated device. This can help you determine data flow errors.

### Monitoring scripts

When your scripts are in working order and are actually performing your test, you can use the Display Monitor Facility to show the current activity of any resource in your simulated network. If you do not use the Display Monitor Facility during a run, you will have to use one of the following procedures:

- Issue Q (Query) operator commands to see what was last transmitted or received, and how far into the script it progressed.
- Insert statements into your scripts to write messages (WTOs) to the operator about the progress of the run.

By monitoring the resources during a run with the Display Monitor Facility, the operator console is not cluttered with WTO messages or query responses.

## Starting the Display Monitor Facility

The following sections describe starting the Display Monitor Facility with the M operator command and logging on to the Display Monitor Facility.

### Using the M operator command

When you start monitoring with the M (Monitor) operator command, the Display Monitor Facility establishes an SNA session with the monitor display. Refer to"M-Display monitor facility" on page 152 for information about the syntax of the M operator command. All data received from the monitoring display is ignored except for information to change the bracket state when Begin Bracket and Change Direction are received. The session remains active until you enter another M operator command to stop monitoring a simulated resource.

### Logging on to the Display Monitor Facility

When you log on to the Display Monitor Facility, the Display Monitor Facility establishes an SNA session with the monitor device and displays a selection panel menu. If you specified the DMUSERPW execution parameter when starting WSim, you are prompted for a password before establishing the SNA session with the monitor display.

After you enter the password, the selection panel menu is displayed. You specify the name of the simulated resource to be monitored and the monitor parameters using the selection panel. After you start monitoring, you can enter PA1 or SNA Signal (ATTN key) to stop monitoring the simulated resource and return to the selection panel. The session remains active until you press the PF3 or PF15 key from the selection panel.

Figure 19 shows the layout of the selection panel.

```
      WSim Version 1 Release 1.0.0 Display Monitor Facility
Name   =                        WSim name of simulated device or 3270 display
View   = SCREEN          DATA or SCREEN - show data stream or screen image
Screen image display only:
Update = XMITRECV        Monitoring display updated when:
                           MONITOR - MONITOR statement is executed from script,
                           TIMER - the specified time value expires, or
                           XMITRECV - data is transmitted/received by display.
Source = BLOCKS          Data stream sent to the monitoring display built from:
                           BLOCKS - internal control blocks
                           DATA - data transmitted/received by display.
Timer = 10               1-600 Seconds when Update = TIMER
Aid   = ON               ON, OFF, or (row,column) location of AID display field

Data stream display only:
Lines = 2                Maximum number of displayed data lines
Code  = EBCDIC           ASCII or EBCDIC - interpret data as ASCII or EBCDIC



ENTER    - Submits parameters to start monitoring of simulated display.
PA1/ATTN - Stops monitoring of simulated display.
PF3/PF15 - Ends Display Monitor Facility session.
```

*Figure 19. Display Monitor Facility selection panel*

## Viewing screen images

To view screen images of a simulated 3270 display, specify the following monitor operands using the M operator command or the selection panel:

- NAME
- VIEW
- UPDATE
- SOURCE
- TIMER
- AID.

### Specifying NAME

The NAME operand specifies the name of the simulated device. For the NAME operand, enter the 1- to 8-character name as specified in the network definition, or the NTWRK name followed by ". " followed by the name of the resource as specified in the network.

### Specifying VIEW

Specify VIEW=SCREEN to have the monitoring display show the screen images of the simulated 3270 display. You cannot specify VIEW=SCREEN to monitor a non-3270 device. However, VIEW=DATA will be automatically selected when you monitor a simulated non-3270 device.

## Specifying UPDATE

The UPDATE operand specifies when the monitoring display is to be updated.

Specify UPDATE=MONITOR to have the monitoring display updated each time a MONITOR statement is executed during the message generation process for the monitored 3270 display. You must specify SOURCE=BLOCKS when you specify UPDATE=MONITOR.

Specify UPDATE=XMITRECV to have the monitoring display updated each time data is transmitted or received by the simulated 3270 display being monitored. You can specify SOURCE=BLOCKS or SOURCE=DATA with UPDATE=XMITRECV.

Specify UPDATE=TIMER to have the monitoring display updated each time the predefined timer value expires. You must specify SOURCE=BLOCKS with UPDATE=TIMER.

## Specifying SOURCE

The SOURCE operand specifies the source used for building the data stream sent to the monitoring display.

Specify SOURCE=BLOCKS to truncate the display image (if necessary) to fit the current display. For example, suppose you are simulating an LU2 defined as a 32 by 132 display. If your display monitor terminal is 24 by 80, coding SOURCE=BLOCKS allows you to see rows 1 to 24 and columns 1 to 80 of the 32 by 132 display image.

**Note:** When you specify SOURCE=BLOCKS, the following occurs:

- Programmed symbols are not shown on the display monitor screen image. However, extended field and character attributes for color and highlighting are shown on the monitoring display.
- Double-byte character set (DBCS) data is displayed when both the simulated and monitor displays support DBCS and have the same screen sizes.
- Field outlining attributes are displayed when the monitor display supports field outlining.
- DBCS data is displayed on a non-DBCS monitor display as single-byte character set data with SO and SI characters shown as < and > respectively.

Specify SOURCE=DATA to not truncate the display image to fit the current display. As a result, attempting to display a simulated 32 by 132 display device's image on a display monitor device defined as a 24 by 80 display may result in an error.

**Notes:**

1. When you specify SOURCE=DATA, your display image is not updated to show the results of the following:
   - An erase input (ERIN) statement is issued by the simulated display.
   - An erase end of field (EREOF) statement is issued by the simulated display.
   - The location of data entered during the message generation process by the simulated display that is preceded by nulls that are suppressed when the message is sent to the host application. The location of the data on the monitor screen is shifted left and overlays the suppressed nulls.
   - Data streams rejected by the simulated display are not passed to the monitor display.

2. When you specify SOURCE=DATA and the simulated display has DBCS=YES specified in the network definition and the monitor display also supports 3270 DBCS data streams, the monitor display may reject data streams for one of the following reasons:
   - The host application program created a DBCS subfield with imbedded nulls that are suppressed if the DBCS subfield is sent back to the host application. In this case, an SI character is not paired with an SO character on the monitor display.
   - Suppressed nulls cause the data sent to the host application program to be shifted left and cause a DBCS character to be split on the monitor display.

If this problem occurs with SOURCE=DATA, use SOURCE=BLOCKS to avoid this situation. As a general rule, use SOURCE=BLOCKS when monitoring simulated 3270 DBCS displays.

### Specifying TIMER

The TIMER operand specifies a timer value in seconds to be used when you specify UPDATE=TIMER.

### Specifying AID

The AID operand specifies whether or not the attention identifier (AID) value generated by the simulated 3270 display is displayed on the monitoring display panel when UPDATE=XMITRECV is specified.

Specify AID=ON to display the AID value on the monitoring display panel in yellow in the lower right corner. Specify AID=OFF to not display the AID value.

Specify AID=(*row*, *column*) to display the AID value on the monitoring display panel in yellow at the specified row and column. You can enter integers from 1 to 255 for the row and column values. If you enter row and column values that cause the AID display field to be displayed on more than one row or to be displayed outside the monitor display panel, the AID display field is moved to the nearest monitoring display panel location.

**Note:** The AID display area is not shown when the simulated device supports double-byte character set data and when you specify SOURCE=DATA.

## Viewing the data stream

To view transmitted and received data streams for a simulated resource, specify the NAME, VIEW, and LINES operands using the M operator command or the selection panel.

### Specifying NAME

The NAME operand specifies the name of the simulated device. For the NAME operand, enter the 1- to 8-character name as specified in the network definition, or the NTWRK name followed by a period (.), followed by the name of the resource as specified in the network.

### Specifying VIEW and LINES

Specify VIEW=DATA to have the monitoring display show the transmitted and received data streams for the simulated device. If you are monitoring a non-3270 display device, VIEW=DATA will be used even if VIEW=SCREEN is specified.

The LINES operand specifies the number of lines shown each time the monitored device transmits or receives data. You can enter a value from 1 to 99 for the LINES operand. If you enter a value greater than the number of rows on the monitoring display, the maximum number available is used. For SNA devices, the minimum number of lines used is two because one of the LINES specified is used for SNA formatting.

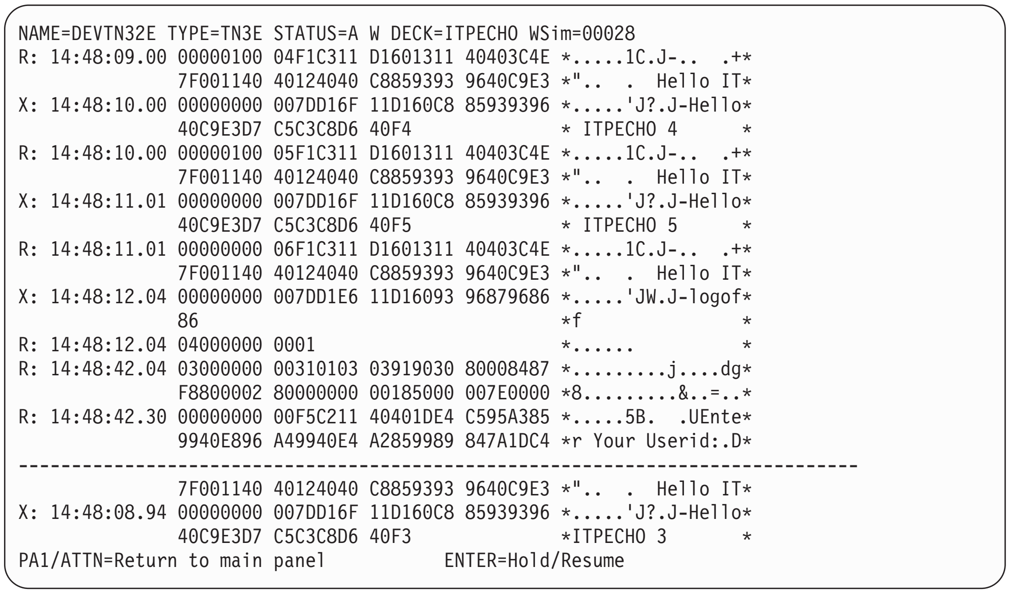
## Specifying CODE

The CODE operand specifies whether the monitored data is to be displayed in ASCII or EBCDIC. EBCDIC is the default.

## Interpreting the Display Monitor Facility data stream display

When you view the data stream, the monitoring display shows the transmitted and received data streams for the simulated device. The number of rows on your screen determine how many rows of data are displayed. Your display screen size must be at least 24 rows by 80 columns to view data streams.

An example of this display is shown in Figure 20.



```
NAME=DEVTN32E TYPE=TN3E STATUS=A W DECK=ITPECHO WSim=00028
R: 14:48:09.00 00000100 04F1C311 D1601311 40403C4E *.....1C.J-..  .+*
               7F001140 40124040 C8859393 9640C9E3 *"..  .  Hello IT*
X: 14:48:10.00 00000000 007DD16F 11D160C8 85939396 *.....'J?.J-Hello*
               40C9E3D7 C5C3C8D6 40F4              * ITPECHO 4      *
R: 14:48:10.00 00000100 05F1C311 D1601311 40403C4E *.....1C.J-..  .+*
               7F001140 40124040 C8859393 9640C9E3 *"..  .  Hello IT*
X: 14:48:11.01 00000000 007DD16F 11D160C8 85939396 *.....'J?.J-Hello*
               40C9E3D7 C5C3C8D6 40F5              * ITPECHO 5      *
R: 14:48:11.01 00000000 06F1C311 D1601311 40403C4E *.....1C.J-..  .+*
               7F001140 40124040 C8859393 9640C9E3 *"..  .  Hello IT*
X: 14:48:12.04 00000000 007DD1E6 11D16093 96879686 *.....'JW.J-logof*
               86                                  *f               *
R: 14:48:12.04 04000000 0001                       *......          *
R: 14:48:42.04 03000000 00310103 03919030 80008487 *.........j....dg*
               F8800002 80000000 00185000 007E0000 *8.........&..=..*
R: 14:48:42.30 00000000 00F5C211 40401DE4 C595A385 *.....5B.  .UEnte*
               9940E896 A49940E4 A2859989 847A1DC4 *r Your Userid:.D*
-------------------------------------------------------------------------------
               7F001140 40124040 C8859393 9640C9E3 *"..  .  Hello IT*
X: 14:48:08.94 00000000 007DD16F 11D160C8 85939396 *.....'J?.J-Hello*
               40C9E3D7 C5C3C8D6 40F3              *ITPECHO 3       *
PA1/ATTN=Return to main panel            ENTER=Hold/Resume
```

*Figure 20. Display Monitor Facility data stream display*

The following fields are shown on this panel:

| | |
|---|---|
| **NAME** | The name of the simulated device being monitored. |
| **TYPE** | The type of the simulated device (sometimes abbreviated). |
| **STATUS** | The status of the simulated device, as indicated in the following list: |

**N** The network is not active.

**A** The device is active.

**L** The device is in an LU-LU session.

**W** Wait indicator is set for the device.

**E** Event wait indicator is set for the device.

**Q** Quiesce indicator is set for the device.

**K** Input is inhibited for the device.

**R** The transaction program is ready.

**T** The transaction program is terminated.

| | |
|---|---|
| **DECK** | The name of the message generation deck or STL procedure that the device is currently executing. |
| **WSIM** | The message generation deck statement number that the device will execute next. |
| **STL** | The STL program statement number (if applicable) that the device will execute next. If there is no STL program being run for this simulation, this field is not displayed. |

## Interpreting data stream messages

Transmitted and received messages are tagged with X (for transmitted messages) and R (for received messages). All messages contain a time stamp indicating when the message was transmitted or received. If your monitor display supports color, these messages are also distinguished by the following colors:

| | |
|---|---|
| **White** | Transmit requests. |
| **Turquoise** | Transmit responses. |
| **Green** | Receive requests. |
| **Pink** | Receive responses. |

Request unit (RU) data is shown in dump format, four 4-byte sets of data with a blank between the sets. A total of 16 bytes of data per line, with translation, is shown. 32 bytes of data per line with translation are shown if your monitor display screen size accommodates it.

Request header (RH) data is displayed before RU data. SNA data streams are formatted, if possible. If an SNA data stream is not defined, the string UNRECOGNIZED is displayed. Sense codes are also formatted, if possible. For non-SNA data streams, no headers appear.

A line of dashes on the monitor display appears below the last message transmitted or received. As messages are displayed, this line moves down the monitoring display and wraps to the top of the display.

**Note:** During heavy message traffic, you may see a red (if your display supports color) line with the message FLUSHED in the middle of a line of equal signs. This means that the simulated device is sending and receiving data faster than the monitor can display the data. When this occurs, the Display Monitor Facility loses messages. (You can use the Loglist Utility to see the lost messages.) You can control the message rate of simulated devices by changing the DELAY and UTI values.

### Controlling the Display Monitor Facility data stream display

You can temporarily suspend and resume monitoring the data stream by pressing **Enter**.

Pressing **Enter** once suspends monitoring the data stream and a red (if your display supports color) line with the message HOLDING replaces the line of dashes. No data stream messages are shown until you resume monitoring.

Pressing **Enter** again resumes monitoring the data stream and a yellow (if your display supports color) line with the message RESUMED replaces the red line. Data stream messages are again shown.

Pressing **Clear** clears the screen and resumes monitoring at the beginning of the data display area. Pressing **Clear** also resumes monitoring if the terminal is in a HOLDING state.

**Note:** All data streams between the time you press **Enter** to suspend monitoring and the time you press **Enter** to resume monitoring are lost.

To return to the Display Monitor Facility Selection Panel, press **PA1** or **ATTN**.

## Specifying BIND profiles

The Display Monitor Facility supports LU Type 0 and LU Type 2 BIND images. Non-SNA 3270 monitor displays use LU Type 0 BIND images and SNA 3270 monitor displays use LU Type 2 BIND images. Other BIND images received during the logon process are turned into valid LU Type 2 BIND images before being sent to the monitoring display.

The BIND image always has one of the following profile fields set into bytes 2 through 7:

| | |
|---|---|
| **LU Type 0** | X'020271402000' |
| **LU Type 2** | X'0303B1903080' |

The monitor display panel size is defined as follows. When VIEW=DATA or VIEW=SCREEN and SOURCE=BLOCKS are specified, the display size fields in the Query Reply are used, if they are available. Otherwise, the display size fields in the BIND image are used.

When VIEW=SCREEN and SOURCE=DATA are specified, the simulated 3270 display panel size is used.

## Chaining to support max RU size when VIEW=SCREEN and SOURCE=DATA

The Display Monitor Facility supports SNA chaining to the monitor display in two ways. First, when the monitor display does not support chaining, the Display Monitor Facility saves the simulated 3270 data as required until an end-of-chain is reached. The data is then sent to the monitor display as an only-in-chain RU truncated to the Max RU Size. When the monitor display supports chaining, the Display Monitor Facility reformats the data as required by the Max RU Size value in the BIND image.

# Chapter 19. Isolating problems

WSim often operates in a complex environment that includes a large physical configuration and many system software components. As a result, you cannot always isolate a problem encountered while operating WSim.

This chapter discusses problem solving in three sections:
- Classifying problems
- Isolating problems
- Reporting problems.

## Classifying problems

You can classify a problem encountered while operating WSim as one of the following:
- Hardware problem
- Software problem
- Problem with WSim installation or procedures.

### Classifying hardware problems

Hardware problems uncovered by WSim are usually related to the host processor running either WSim or the system under test.

### Classifying software problems

Software problems may be related either to WSim or to the system under test. WSim problems can occur in either:
- WSim host processor code
- The operating system.

Problems in the system under test can occur in either:
- Operating system
- Application programs being tested.

Errors in other system software components are not discussed here because one function of WSim is to discover such errors.

### Classifying problems with installation and procedures

Problems you encounter when operating WSim may result from incorrect system setup or a failure to follow established WSim procedures. These problems often prevent WSim from operating or severely limit WSim functions.

Examples of problems with installation and procedures include the following:
- Invalid statement sequence in the message generation deck
- Invalid logic test definition.

## Isolating problems

This section discusses the types of problems that may involve WSim and gives suggestions for isolating and solving the problems. If the methods fail to produce a solution, you should assume that the problem is caused by a WSim error and follow the procedure for formally reporting a WSim problem given in“Problem reporting” on page 135.

### Program checks

A program check that occurs in the host processor code is probably a WSim problem. However, a program check in the Loglist Utility, the Log Compare Utility, or the Response Time Utility may be caused by an attempt to process a data set other than a WSim log data set. This error may also occur if WSim does not write the end-of-file indicator (tape mark on a tape data set) because of system failures. The most common program check in this situation is a data exception (program interruption code 7 or 0C7 ABEND), but others are also possible. You should obtain documentation on the problem and report it using the procedure in“Problem reporting” on page 135.

### Loops

If the host processor task remains in a system state and there is no activity on the lines, then WSim is probably in a program loop. WSim program loops can be caused by specifying incorrect statement sequences in your message generation decks. Verify that the message generation decks in your networks contain delimiter statements as discussed in *Creating WSim Scripts*. Also, verify your STL programs do not contain any loops that can never be exited. To isolate a loop problem, run WSim with MSGTRACE=YES in your network definition, issue the ZEND operator command, and run the Loglist Utility to determine the loop path. If the message generation decks or STL programs are correct, then the loop is probably a WSim problem. Record some of the storage addresses in the loop, make a storage dump of the WSim region, and report the problem.

### Incorrect or missing message traffic

If you are running a network, and no message traffic is on the lines or incorrect data is being transmitted or received, one of the following problems may exist:

- Large intermessage delays: Cancel the problem network and then initialize and start it with a network-level UTI value of 0. If message traffic starts almost immediately, your intermessage delay values may be excessively large.
- Invalid response from the system under test: Verify that the system under test is sending the correct data. This can be very important in the case of SNA header information and 3270 buffer control characters.
- Network logic control: The following suggestions may be helpful in determining why a terminal or device is waiting or generating incorrect data.
  - If your network logic is complicated, you may want to generate a small, simple test network for debugging purposes.
  - Verify that the message generation decks or STL programs for the device do not contain a misplaced WAIT statement.
  - Verify that the terminals are not waiting for a message from the application program that has not been sent.
  - Use the console recovery facility to reset the WAIT indicator for the device and to generate a message for the device from the operator console.
  - For message generation decks, check the IF statements for the device.

1. Verify that the LOC or LOCTEXT operand value is correct.
2. Verify that the TEXT or AREA operand value is correct.
3. Verify that a previous IF statement with STATUS=HOLD is not causing the problem.
4. Verify that a network-level IF statement is not causing the problem.
5. Verify that the actions in a series of IF statements do not contradict one another.

– For STL programs, check the following:
  1. Verify that the ONIN, ONOUT, and WAIT statement conditions are correct.
  2. Make sure that WAIT statements are not coded without the UNTIL clause.
– Use the MSGTRACE network option to help determine the cause of logic problems. This option allows you to follow the steps of message generation. It helps you to determine if you are actually sending messages when you expect to, if the proper IF statements are activated when the logic test is processed, when calls and branches are made, and when EVENTs are signaled, posted, or reset.
– Use the STLTRACE network option to help trace the messages of a simulation, but at the higher STL program level. This option allows you to follow the steps of execution through your STL program.

- Loops: See“Loops” on page 134 for information.
- Wrong message generation decks, STL programs, or networks: Initialize your network with the LIST option. This option lists the network, message generation decks, and STL programs you are running.

### No outstanding WTOR

If you are using the WTOR command interface, the absence of a WTOR prompt may be due to one of the following reasons:

- WSim subtask ABEND
- Loop in WSim
- Problem with the operating system.

If you cannot establish that the problem is with the operating system, the loss of the WTOR is probably a WSim problem and should be reported.

## Problem reporting

If you have encountered a problem with WSim and have not been able to resolve it by following the procedures given in this chapter, you should open a PMR on the RETAIN* system and have the following information available:

- A dump of the WSim region in the host processor
- Listings of your networks and message generation decks. Initialize your network with the LIST option. This will list the exact network and message generation decks that you are running.
- The operator console listing
- A complete listing of the log data set or the data set itself
- Dispatcher trace (if applicable)
- A dynamic dump of the control blocks (if applicable)
- A description of the environment at the time of the problem.

# Chapter 20. Specifying operator commands

The following sections provide a list of the operator commands, specification operands, and operation operands you can use to control simulations. To help you make use of this information, the following sections explain the terminology and the conventions used when describing the operator commands.

## Understanding operator command coding terms

For the various operator commands discussed in this chapter, the following terms will be used:

| | |
|---|---|
| ***ntwrk*** | Indicates that you should enter a network name. This is the 1- to 8-character symbol in the name field of the NTWRK statement. |
| ***tcpip*** | Indicates that you should enter a name of a TCP/IP group. This is the 1- to 8-character symbol in the name field of the TCPIP statement that defines a TCP/IP connection. |
| ***dev*** | Indicates that you should enter a device name. This is the 1- to 8-character symbol in the name field of the DEV statement. |
| ***lu*** | Indicates that you should enter an LU name. This is the 1 to 8-character symbol in the name field of a VTAMAPPL simulation LU statement. |
| ***vtamappl*** | Indicates that you should enter a VTAM application (VTAMAPPL) name. This is the 1- to 8-character symbol in the name field of a VTAMAPPL statement. |
| ***tp*** | Indicates that you should enter a transaction program (TP) name. This is the 1- to 8-character symbol in the name field of a TP statement in a CPI-C simulation. |
| ***appclu*** | Indicates that you should enter the name of an APPC LU. This is the 1- to 8-character symbol in the name field of an APPCLU statement in a CPI-C simulation. |

## Understanding operator command coding conventions

These conventions are used in the following operator command descriptions:

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

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

  A conditional operand is one that you may need to code or to omit, depending on how you code (or omit) other operands in the command. For each conditional operand, the conditions under which you should code or omit it are indicated.
- Braces, { }, and vertical lines, |, indicate that an operand has a value which you must choose from the alternatives listed.
- An ellipsis (...) indicates that you may code a sequence of values within parentheses.

## Understanding resource names

The person creating the NTWRK generation deck gives each resource in the simulation a name by which the resource can be correlated with operator commands, logs, and traces. Clearly, if only one NTWRK is being simulated at a time, the resource name is in the NTWRK being used. The resource name alone is sufficient to uniquely identify the resource.

However, some simulation environments involve more than one NTWRK. In those situations it may be desirable for a resource name to be duplicated in the different NTWRKs. WSim allows you to qualify a resource name with the NTWRK name, to form a *NTWRK-qualified resource name*. A NTWRK-qualified resource name is formed by a NTWRK name, followed by a period, followed by the resource name. For example, TESTLU in the NTWRK named network2 would have a NTWRK-qualified resource name of network2.TESTLU.

## Using operator commands

The following section describes the operator commands, specification operands, and operation operands you can use to control simulated resources.

### A-Alter network operands

```
A [ resource,]{ area=newtext}
```

```
           {ATRABORT={DECK|PATH|NONE}}
           {ATRDECK= name}
           {CONRATE={YES|NO}
           {CPITRACE={MSG|VERB|VERBEND|NONE}}
           {D=dname,sname}
           {DC n= integer}
           {DEBUG={ON|OFF}
           {DSEQ= integer}
           {E=[rate][, interval]}
           {H=chars}
           {I= integer}
           {IUTI=utiname}
           {LC n=integer}
           {LOGDSPLY={BEGIN|BOTH|END|NONE}}
           {LSEQ= integer}
           {M={A( integer)|F( integer)|R(n1[, n2])|T(integer)}
           {MONCMND={ON|OFF}}
           {MSG n={ON|OFF}}
           {MSGTRACE={YES|NO}}
           {NC n= integer}
           {NSEQ= integer}
           {NSW[n]={ON|OFF}}
           {PATH={pathname|( pathname,...)}}
           {PORT=portnumber}
           {POST=event}
           {PRTSPD=integer}
           {QUIESCE}
           {R= dname}
           {RELEASE}
           {RESET=event}
           {RSTATS}
           {S=[ x][,[ y][,{ z|OFF}]]}
           {SERVADDR=addr}
           {SIGNAL= event}
           {STLTRACE={YES|NO}}
           {SW[n]={ON|OFF}}
           {TCn=integer}
           {TSEQ= integer}
           {TSW[ n]={ON|OFF}}
           {U= integer}
           {UTI=(utiname,integer)}
           {X={FULL|LINE|RATE|NONE}}
```

This command alters operand values for specified network resources.

*resource*

**Function:** This specifies the resource to be altered.

**Format:** You can specify one of the following values for *resource*:

| | |
|---|---|
| *ntwrk* | Indicates an alteration of the entire specified network. |
| *tcpip* | Indicates an alteration of all devices in a TCP/IP group. |
| *dev* | Indicates an alteration of the specified device. *dev* should be a unique name for all active networks. |
| *lu*[-*num*] | Indicates an alteration of the specified LU or a session associated with that LU. -*num* specifies the session number of the LU session to be altered. If you do not specify - *num*, all sessions associated with the LU are altered. *lu* should be a unique name for all active networks. *lu-** is also acceptable, and resolves to the last session number. |
| *vtamappl* | Indicates an alteration of all LUs using the specified VTAMAPPL. |

| | |
|---|---|
| *tp*[-*num*] | Indicates an alteration of the specified CPI-C transaction program (TP). -*num* specifies the TP instance number to be altered. This operand only applies to TPs that support multiple instances. If the operand is specified for a TP that does not support multiple instances, it is ignored. -*num* is a decimal number in the range from 1 to 99999. If -*num* is omitted, the command applies to the first TP instance. If *tp*[-*num*] is specified without the *appclu* qualifier, the TP name must be unique within the network. *tp*-* is also acceptable, and resolves to the last instance number. |
| *appclu* | Indicates an alteration of all TPs defined on the specified APPCLU. |
| *appclu.tp*[-*num*] | Indicates an alteration of the specified CPI-C transaction program (TP). -*num* specifies the instance number of the TP to be altered. If you do not specify - *num*, the first instance is altered. *appclu.tp*-* is also acceptable and resolves to the last instance. |

Table 2 shows which A (Alter) command operands are valid with the indicated levels of network resources.

*Table 2. Valid A (Alter) command operands for network resource levels*

| Operand | All Simulations | CPI-C Simulation or VTAMAPPL Simulation | TCP/IP Simulation |
|---|---|---|---|
| U± *integer* | All*, NTWRK | VTAMAPPL, LU, APPCLU, TP | TCPIP, DEV |
| s | All*, NTWRK | VTAMAPPL, LU, APPCLU, TP | TCPIP, DEV |
| N± *integer* | All*, NTWRK | None | None |
| N *s* | All*, NTWRK | None | None |
| ATRABORT | All*, NTWRK | VTAMAPPL, LU | TCPIP, DEV |
| ATRDECK | NTWRK | VTAMAPPL, LU | TCPIP, DEV |
| CONRATE | All*, NTWRK | None | None |
| CPITRACE | All*, NTWRK | APPCLU, TP | None |
| D | NTWRK | None | None |
| DC*n* | All*, NTWRK | VTAMAPPL, LU, APPCLU, TP | TCPIP, DEV |
| DEBUG | All*, NTWRK | None | None |
| DSEQ | All*, NTWRK | VTAMAPPL, LU, APPCLU, TP | TCPIP, DEV |
| E | All*, NTWRK | None | None |
| H | All*, NTWRK | None | None |
| I | All*, NTWRK | None | None |
| IUTI | NTWRK | VTAMAPPL, LU, APPCLU, TP | TCPIP, DEV |
| LC *n* | All*, NTWRK | VTAMAPPL, APPCLU | TCPIP |
| LOGDSPLY | All*, NTWRK | VTAMAPPL, LU | TCPIP, DEV |
| LSEQ | All*, NTWRK | VTAMAPPL, APPCLU | TCPIP |
| M | NTWRK | VTAMAPPL, LU | TCPIP, DEV |
| MONCMND | All*, NTWRK | None | None |
| MSG *n* | All*, NTWRK | None | None |

Table 2. Valid A (Alter) command operands for network resource levels (continued)

| Operand | All Simulations | CPI-C Simulation or VTAMAPPL Simulation | TCP/IP Simulation |
|---|---|---|---|
| MSGTRACE | All*, NTWRK | VTAMAPPL, LU, APPCLU, TP | TCPIP, DEV |
| NC *n* | All*, NTWRK | None | None |
| NSEQ | All*, NTWRK | None | None |
| NSW *n* | All*, NTWRK | None | None |
| PATH | NTWRK | VTAMAPPL, LU, APPCLU, TP | TCPIP, DEV |
| PORT | All*, NTWRK | None | TCPIP, DEV |
| POST | NTWRK | None | None |
| PRTSPD | All*, NTWRK | VTAMAPPL, LU | None |
| QUIESCE | All*, NTWRK | VTAMAPPL, LU, APPCLU, TP | TCPIP, DEV |
| R | NTWRK | None | None |
| RELEASE | All*, NTWRK | VTAMAPPL, LU, APPCLU, TP | TCPIP, DEV |
| RESET | NTWRK | None | None |
| RSTATS | All*, NTWRK | VTAMAPPL, LU | TCPIP, DEV |
| S | All*, NTWRK | None | None |
| SERVADDR | All*, NTWRK | None | TCPIP, DEV |
| SIGNAL | NTWRK | None | None |
| STLTRACE | All*, NTWRK | VTAMAPPL, LU, APPCLU, TP | TCPIP, DEV |
| SW*n* | All*, NTWRK | VTAMAPPL, LU, APPCLU, TP | TCPIP, DEV |
| TC *n* | All*, NTWRK | VTAMAPPL, APPCLU, TP | TCPIP |
| TSEQ | All*, NTWRK | VTAMAPPL, APPCLU, TP | TCPIP |
| TSW*n* | All*, NTWRK | VTAMAPPL, APPCLU, TP | TCPIP |
| U | All*, NTWRK | None | None |
| UTI | NTWRK | None | None |
| X | All*, NTWRK | None | None |

* Applies to all active networks

**Default:** If you do not enter a value for *resource*, all networks are altered.

*area= newtext*

**Function:** Specifies the save area or user area and the new text information that is to be stored in the save area or user area.

**Format:** You can specify the following values for the *area* operand:

| | |
|---|---|
| **N±** *integer* | Indicates the offset within the network user area where *newtext* is inserted. *integer* is an integer between 0 and 32766 and must be specified. |

| | |
|---|---|
| **U±** *integer* | Indicates the offset within the device user area where *newtext* is to be inserted. *integer* is an integer between 0 and 32766 and must be specified. |
| **N** *s* | Indicates the network save area where *newtext* is to be inserted. *s* is an integer between 1 and 4095. You cannot specify offsets for network save areas. |
| *s* | Indicates the device save area where *newtext* is to be inserted. *s* is an integer between 1 and 4095. You cannot specify offsets for device save areas. |

*newtext* is the text to be stored in the user area or save area.

**Notes:**

- You can specify *newtext* as any combination of EBCDIC and hexadecimal data. The hexadecimal portion of the data must be enclosed within single quotes. To specify an EBCDIC quote as part of the data, you must specify two continuous quotes.
- EBCDIC data entered is not folded to uppercase.
- You can enter literal text DBCS data for *newtext*.
- If a value is valid and causes *newtext* to extend past the end of the area, the alteration is performed, truncating the overflow.

**ATRABORT={DECK|PATH|NONE}**

**Function:** Specifies whether the current message generation deck or current path of a device is to be ended when automatic terminal recovery (ATR) is invoked. Refer to *Creating WSim Scripts* for a discussion of ATR.

**ATRDECK=***name*

**Function:** Specifies whether the current message generation deck or current path of a device is to be ended when automatic terminal recovery (ATR) is invoked. Refer to *Creating WSim Scripts* for a discussion of ATR.

**CONRATE={YES|NO}**

**Function:** Specifies whether the interval report rates are to be printed at the operator console.

**CPITRACE={MSG|VERB|VERBEND|NONE}**

**Function:** 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 dataset for formatting by the loglist program.

**Format:** You can specify the following values for the CPITRACE operand:

| | |
|---|---|
| **MSG** | Specifies that CPI-C trace messages are to be logged, indicating the issuance and completion of CPI-C verbs. |
| **VERB** | Specifies that CPI-C verbs are to be logged when they are issued and when they complete. |
| **VERBEND** | Specifies that CPI-C verbs are to be logged when they complete. |
| **NONE** | Specifies that no CPI-C trace information is to be logged. **Note:** If CPITRACE=NONE is specified, all CPI-C message logging is inhibited, including error messages. |

**D=** *dname*,*sname*

**Function:** Specifies deletion and substitution of a message generation deck or STL procedure. This function specifies that the message generation deck or STL procedure *dname* is to be marked not usable (deleted) and the message generation deck or STL procedure *sname* will be substituted for the deleted deck or procedure. *sname* must not be currently deleted. When a new path entry is selected and you have used this command to delete the message generation deck or STL procedure named in that entry, the substitute message generation deck or STL procedure (*sname*) will be used instead of *dname* as the new path entry. Similar substitutions are made for the decks or procedures named by the FRSTTXT and ATRDECK operands of the NTWRK, TCPIP, DEV, VTAMAPPL, or LU statements (or by the FRSTTXT operand of the APPCLU or TP statement), if these decks or procedures have been deleted at the time they are invoked. If the deck or procedure is branched into or called from other message generation decks or STL procedures, this deletion has no effect. The substitution is made only when the deleted message generation deck or STL procedure is selected by a PATH statement or the FRSTTXT or ATRDECK operands.

**DC***n*= *integer*

**Function:** Specifies that device index counter *n* is to be set to the provided *integer* value.

**Format:** *n* is from 1 to 4095 and *integer* is from 0 to 2147483647.

**DEBUG={ON|OFF}**

**Function:** Specifies that the network DEBUG option is to be turned on or off. This function logs SNA request units containing encrypted or translated data.

**DSEQ=** *integer*

**Function:** Specifies that the device sequence counter is to be set to *integer*

**Format:** *integer* is from 0 to 2147483647.

**E=[***rate***][,** *interval***]**

**Function:** Specifies a change in the expected message transfer rate (messages transmitted from WSim) and the adjustment interval through which WSim automatically adjusts the UTI. Refer to *Creating WSim Scripts* for information on the automatic UTI adjustment facility.

**Note:** For networks that have multiple UTIs defined, the E operand will adjust all UTIs in the network.

**Format:** The value for *rate* specifies the expected rate in messages per minute. It is an integer from 0 to 65535. The value for *interval* specifies the adjustment interval in seconds. It is an integer from 0 to 65535.

**Default:** If *rate* is omitted, the previously specified rate is left unchanged.

**H=** *chars*

**Function:** Specifies the new network heading text for interval reports. Format: *chars* can be 1 to 24 bytes of data. You can enter literal text DBCS data. The data is not folded to uppercase.

**Default:** If you do not enter data, the heading is set to blanks.

**Note:** You can use this operand on a network level only.

**I=** *integer*

**Function:** Specifies that the time between printing interval reports is to be altered.

**Format:** *integer* represents minutes and may be from 1 to 240.

**Note:** You can use the I operand to save paper. Set I=240 initially. When an interval report is required, set I=1 and, after the 1-minute interval has expired, you receive the report. Reset I=240 until another report is required.

**IUTI=** *utiname*

**Function:** Specifies that the individual UTI for the specified device, terminal, TP, or LU is to be changed to *utiname*.

**Format:** If IUTI=NTWRKUTI is specified, the network UTI is selected for the individual UTI for the device.

**LC** *n***=** *integer*

**Function:** Specifies that line index counter *n* is to be set to *integer*.

**Format:** *n* is from 1 to 4095 and *integer* is from 0 to 2147483647.

**LOGDSPLY={BEGIN|BOTH|END|NONE}**

**Function:** Specifies whether display and printer buffers are to be automatically logged for formatting by the Loglist Utility. This option is valid for simulated 3270 and 5250 terminals and 3270 printers. It will be ignored for all others.

**Format:** You can specify the following values for the LOGDSPLY operand:

| | |
|---|---|
| **BEGIN** | Indicates that the display is to be logged at the beginning of message generation only. |
| **BOTH** | Indicates that the display is to be logged both at the beginning and end of message generation for the device. |
| **END** | Indicates that the display is to be logged at the end of message generation only. |
| **NONE** | Indicates that there will be no automatic display logging. |

**LSEQ=** *integer*

**Function:** Specifies that the line sequence counter is to be set to *integer*.

**Format:** *integer* is from 0 to 2147483647.

**M={A(***integer***)|F(***integer***)|R(***n1***[,** *n2***])|T(***integer***)}**

**Function:** Specifies the value multiplied by the user time interval (UTI) to define the delay between messages to be altered. Refer to the DELAY operand description under the LU (VTAMAPPL) statement in *WSim Script Guide and Reference*.

**Format:** You can specify the following values for the M operand:

| | |
|---|---|
| **A** | Indicates that the delay is an average chosen randomly in the range from zero to two times the specified integer. The value *integer* may be from 0 to 1073741823. The average delay will be an integer value. |
| **F** | Indicates that the delay is fixed and the integer specifies the value. The value *integer* may be from 0 to 2147483647. |
| **R** | Indicates that the delay is to be random, within the specified boundaries. The value *n1* is an integer from 0 to 255 and specifies the number of the RN statement that names the rate table used in selecting the delay.<br>If you specify two values, the values *n1* and *n2* are integers from 0 to 2147483647 and specify the low (*n1*) and high (*n2*) limits of the range of random delay values. *n1* must be less than *n2*. |

**T** Indicates that the delay will be chosen randomly from the rate table on the referenced RATE statement. The value *integer* specifies the name field on a RATE statement and may be from 0 to 255.

**MONCMND={ON|OFF}**

**Function:** Specifies whether the function of monitoring script operator commands and their responses is to be turned on or off. ON indicates that all operator commands generated by message generation deck OPCMND statements and their responses are monitored at the operator console.

**MSG** ***n*****={ON|OFF}**

**Function:** Specifies whether message number *n* is to be displayed to the console and logged to the log data set.

**Format:** *n* is any message number that can be inhibited. Refer to *WSim Script Guide and Reference* for more information about the INHBTMSG operand on the NTWRK statement.

**MSGTRACE={YES|NO}**

**Function:** Specifies whether message generation trace records should be written to the log data set. If you specify a network, all resources within that network will be affected. Otherwise, only the specified resource is affected.

**NC** ***n*****=** *integer*

**Function:** Specifies that network index counter *n* is to be set to the value *integer*.

**Format:** *n* is from 1 to 4095 and *integer* is from 0 to 2147483647.

**NSEQ=** *integer*

**Function:** Specifies that the network sequence counter is to be set to the value *integer*.

**Format:** *integer* is from 0 to 2147483647.

**NSW[*****n*****]={ON|OFF}**

**Function:** Specifies that network-level switch *n* is to be set on or off.

**Format:** *n* is an integer from 1 to 4095.

**Default:** If you do not specify a value for *n*, all network-level switches are to be set on or off.

**PATH={** *pathname***|(***pathname***,...)}**

**Function:** PATH= *pathname* specifies the PATH statement to be executed before the next normally selectable PATH. This function occurs only once for each path insertion entered. Normal PATH processing is resumed after the inserted PATH is complete.

PATH=(*pathname*,...) specifies the new PATH statements to be referenced by the terminals or devices. If you change a path using the A (Alter) command, the R (Reset) command does not restore the previously defined path. You can restore a path only by specifying the original path on another A (Alter) command. If you specify PATH=(), the current path for the terminal or device is deleted and processing continues based upon the sequence of PATH statements.

**Format:** *pathname* is the name coded in the name field on a PATH statement in the network definition. If you specify multiple PATHs, each *pathname* is separated by commas and the entire field enclosed in parentheses. You can enter a maximum of either 25 path entries or 120 characters, including the A (Alter) command and qualification.

**PORT=** *portnumber*

**Function:** Specifies the local port number to be used by a Simple TCP or Simple UDP device.

**Format:** *portnumber* is an integer from 1 to 65535 representing the local port number to be used.

**POST=** *event*

**Function:** Specifies the name of an event that is to be posted complete.

**Format:** *event* is from 1 to 8 alphanumeric characters.

**PRTSPD=** *integer*

**Function:** Specifies that the print speed is to be altered.

**Format:** *integer* is from 0 to 32767 and represents characters per second.

**QUIESCE**

**Function:** Specifies that the network, TCP/IP connections, VTAMAPPLs, APPCLUs, LUs, or TPs are to receive messages and respond negatively to polls while they are not generating any data messages.

**R=** *dname*

**Function:** Specifies the reinstatement of a previously deleted message generation deck or STL procedure. The message generation deck or STL procedure named in this operand is to be reinstated as usable. The substituted message generation deck or STL procedure (*sname*), determined by the D (Delete) operand on the A (Alter) command, will no longer be selected for the reinstated deck or procedure (*dname*).

**RELEASE**

**Function:** Specifies that the network, TCP/IP connections, VTAMAPPLs, APPCLUs, LUs, or TPs are to be released from the quiesce state.

**RESET=** *event*

**Function:** Specifies the name of an event that is to be marked as not complete.

**Format:** *event* is from 1 to 8 alphanumeric characters.

**RSTATS**

**Function:** Resets the online response-time statistics (RSTATS) for the specified resources to zero. This prevents previous statistics from biasing future figures and could be especially useful when you alter UTI values or change test system parameters during a run.

**Note:** You can also reset online response-time statistics with the R (Reset) operator command.

**S=[***x***][,[***y***][,{** *z***|OFF}]]**

**Function:** Specifies that the SCAN values are to be altered. Refer to the description of the SCAN operand under the NTWRK statement in *WSim Script Guide and Reference* for more information.

**Format:** You can specify the following values for the S operand:

| | |
|---|---|
| *x* | The report interval and is between 0 to 255. |
| *y* | The threshold value and is between 0 to 255. |
| *z* | The delay before invoking automatic terminal recovery. The limits are 0 to 255. This also activates automatic terminal recovery. |
| **OFF** | Deactivates automatic terminal recovery. |

**Default:** If you omit any of the operands, the corresponding values for the network are left unchanged.

**SERVADDR=** *addr*

**Function:** Specifies the host server address to which you want to connect.

**Format:** *addr* is a dotted decimal IP address.

**SIGNAL=** *event*

**Function:** Specifies the name of an event to be signaled.

**Format:** *event* is from 1 to 8 alphanumeric characters.

**STLTRACE={YES|NO}**

**Function:** Specifies whether or not STL trace records should be written to the log data set. If you specify a network, all resources within that network will be affected. Otherwise, only the specified resource is affected.

**SW[*n*]={ON|OFF}**

**Function:** Specifies that device switch *n* is to be set on or off.

**Format:** *n* is an integer from 1 to 4095.

**Default:** If you do not specify a value for *n*, all device switches are to be set on or off.

TC *n*= *integer*

**Function:** Specifies that terminal index counter *n* is to be set to the value *integer*.

**Format:** *n* is from 1 to 4095 and *integer* is from 0 to 2147483647.

**TSEQ=** *integer*

**Function:** Specifies that the terminal sequence counter is to be set to the value *integer*.

**Format:** *integer* is from 0 to 2147483647.

**TSW[*n*]={ON|OFF}**

**Function:** Specifies that the terminal-level switch indicated by *n* is to be turned on or off.

**Format:** *n* is an integer from 1 to 4095.

**Default:** If you do not specify a value for *n*, all terminal level switches are to be turned on or off.

**U=** *integer*

**Function:** Specifies that the UTI defined on the NTWRK statement is to be changed to the indicated value.

**Format:** *integer* can be an integer from 0 to 65535.

**UTI=(***utiname***,***integer***)**

**Function:** Specifies that the individual UTI *utiname* is to be changed to the value of *integer*.

**Format:** *utiname* can be from 1 to 8 alphanumeric characters. *integer* can be an integer from 0 to 65535.

**X={FULL|LINE|RATE|NONE}**

**Function:** Specifies that the type of interval report should be changed to the option specified.

**Format:** You can specify the following values for the X operand:

| | |
|---|---|
| **FULL** | Specifies that the entire report, including terminal and device statistics, line totals, cumulative totals, and rates, is printed. |
| **LINE** | Specifies that the report includes only line totals, cumulative totals, and message rates. |
| **RATE** | Specifies that the report includes only the network totals and message rates. |
| **NONE** | Specifies that no interval report is printed. |

## C-Cancel network resources

```
C [resource]
```

This command cancels a network or an APPCLU, VTAMAPPL, or TCPIP within the network. If a network is canceled, an end of run report will be printed and all control blocks for the network will be deleted from the system. To begin using the canceled resource again, you must initialize the network.

*resource*

**Function:** Specifies the resource to be canceled.

**Format:** You can specify the following values for *resource*:

| | |
|---|---|
| *ntwrk* | Specifies a single network for which all resources are to be canceled. |
| *appclu* | Specifies an APPCLU to be canceled. |
| *vtamappl* | Specifies a VTAMAPPL to be canceled. |
| *tcpip* | Specifies a TCP/IP connection to be canceled. |

**Default:** If you do not specify a value for *resource*, all networks are canceled.

## D-Dump control blocks

```
D ntwrk,{L[=resource]|N}
```

This command dumps the contents of control blocks to the printer.

The output produced by this command is provided as a source of information to help with debugging WSim problems. This internal information should never be used as programming interface information.

*ntwrk*

**Function:** Specifies the network for which control blocks are to be dumped.

**{L[=** *resource***]|N}**

**Function:** Specifies which resources are to be dumped.

**Format:** You can specify the following values for this operand:

| | |
|---|---|
| **L** | Specifies that all TCP/IP connections and their associated control blocks are to be dumped. |
| **L**= *tcpip* | Specifies that the TCP/IP connection control block for the specified *tcpip* is to be dumped. |
| **N** | Specifies that the Network Control Block (NCB) and associated message generation control blocks are to be dumped. |

**Default:** If you omit the L and N operands, all control blocks in the network are dumped.

## E-Restart message logging

```
E [ ntwrk]
```

The E command restarts message logging for the specified *ntwrk* log data set (NTWRKLOG) or for the general log data set if you do not specify *ntwrk*.

For more information about logging by network, refer to *Creating WSim Scripts*. For more information about the NTWRKLOG statement, refer to *WSim Script Guide and Reference*.

## F-Enter console recovery

```
F resource
```

This command places the specified resource in console recovery mode where you can transmit specific sequences or end the current PATH entry. Console message ITP100I DATA RECEIVED - *data* is issued to indicate any data received while in console recovery mode. You can use the subcommands described below with the information from message ITP100I to create an interactive conversation between the operator and the host application.

Entering console recovery causes all wait conditions, delays, and input inhibited conditions to be reset. It is possible, therefore, that a terminal can be recovered by simply entering and exiting console recovery.

*resource*

**Function:** Specifies the resource for console recovery.

**Format:** You can specify the following values for *resource*:

| | |
|---|---|
| *dev* | Specifies the device for recovery. |
| *lu*[- *num*] | Specifies the LU session for recovery. *-num* indicates the session within the LU. If you omit *-num*, the recovery is started for the first session within the LU. *lu*-* can also be specified, which resolves to the last session number. |

While in this mode, you can enter only console recovery subcommands. These subcommands define the action to be taken. The subcommands are as follows:

**Subcommand** **Description**

* *data*
: This subcommand is an asterisk followed by data. It causes the specified data (through the last nonblank) to be entered as a message from the terminal or device in console recovery mode. An asterisk entered without data generates:
- An Enter key for 3270 and LU7
- An unpredictable 1-character message for all other devices.

**Notes:**
- You can specify *data* as EBCDIC and hexadecimal. The hexadecimal portion of the data must be within single quotes. To specify an EBCDIC quote (') as part of the data, you must specify two continuous quotes ('').
- The EBCDIC data entered is not folded to uppercase.
- You can enter literal text DBCS data.

**K** *key* **[(***c***)]**
: This subcommand simulates the action of the program function keys of 3270 or LU7 devices, where *c* is an optional data character to be entered at the current cursor position before simulating the action of the key specified. It is valid for these device types only. Keys supported for 3270 are:

ENT CLR PA1 PA2 PA3 PF1 PF2 PF3 ... PF24

Keys supported for LU7 are:

ENT CLR CMD1 CMD2 CMD3 ... CMD24

**N**
: This subcommand ends message generation using the current message generation deck. Message generation begins at the next specified message generation deck. The command also ends console recovery mode.

**Q**
: This subcommand enables you to query the specified resource. The resulting display is the same information reported for a Q (Query) operator command for the resource that is in console recovery.

**X**
: This subcommand ends console recovery mode without further changes to message generation.

## G-Terminal status query

```
G [ntwrk],{ACT|INACT|QUIESCE|READY|TERM}
```

This command displays those simulated terminals which are active, inactive, quiesced, ready, or terminated. For this command, *terminal* means any message generating resource.

`ntwrk`

**Function:** If multiple networks are being simulated, *ntwrk* specifies the network to which this command pertains.

**Default:** If you do not specify a value for *ntwrk*, all terminals in all initialized networks are listed.

**`{ACT|INACT|QUIESCE|READY|TERM}`**

**Function:** Specifies the status type to be listed.

**Format:** You can specify one of the following values for this operand:

| | |
|---|---|
| **ACT** | Lists active terminals. |
| **INACT** | Lists inactive terminals. |
| **QUIESCE** | Lists quiesced terminals. **Note:** Terminals that are quiesced are not listed as active. |
| **READY** | Lists ready terminals. This status only applies to TP instances in a CPI-C simulation. This status indicates that the TP instance is a server that is waiting for an incoming attach request. |
| **TERM** | Lists terminated terminals. This status only applies to TP instances in a CPI-C simulation. This status indicates that all message generation activity defined for this TP instance has completed. |

## I-Initialize a network

```
Intwrk[,{LN|L}]
      [,s]
```

This command causes WSim to read the network definition statements from the specified member of the data set defined by the INITDD DD statement. The member has the same name as the name field of the NTWRK statement. The members specified on INCLUDE statements in the network are read from the data set defined by the MSGDD DD statement. After all members have been read, the necessary control blocks are built for the network simulation.

`ntwrk`

**Function:** Specifies the data set member name of the network to be initialized.

**`{L|LN}`**

**Function:** Specifies what is to be printed when the network is initialized.

**Format:** You can specify the following values for this operand:

| | |
|---|---|
| **LN** | Specifies that only the network definition statements are to be listed on the printer as the network is initialized. |
| **L** | Specifies that the entire script (network definition and message generation statements) is to be listed on the printer as it is initialized. |

**`S`**

**Function:** Specifies that the network is to be automatically started at the completion of initialization.

## M-Display monitor facility

```
M [ resource,]{ON, DISPLAY=name|OFF}
           [,AID={ON |OFF|(row,column)}
           [,CODE={EBCDIC |ASCII}
           [,DLOGMOD=logname
           [,LINES={number|2}]
           [,SOURCE={ BLOCKS |DATA}
           [,TIMER={integer|10}]
           [,UPDATE={MONITOR|TIMER|XMITRECV}]
           [,VIEW={DATA|SCREEN}]
```

This command causes the Display Monitor Facility to start or stop a session with a monitor display, or list the devices being monitored. Each Display Monitor Facility session is associated with the simulated device being monitored.

*resource*

**Function:** Specifies the simulated device or simulated LU session to start or stop monitoring.

**Format:** You can specify the following values for *resource*:

| | |
|---|---|
| *dev* | Specifies the simulated device to start or stop monitoring. |
| *lu*[-*num*] | Specifies the simulated LU session to start or stop monitoring, where -*num* indicates which half-session of this LU is to be monitored. If you omit -*num*, the first session is assumed. *lu*-* can also be specified, which resolves to the last session number. |
| *tp*[-*num*] | Specifies the simulated CPI-C transaction program to start or stop monitoring, where -*num* indicates which instance of this TP is to be monitored. If you omit -*num*, the first instance is assumed. *tp*-* can also be specified, which resolves to the last instance number. |

**Default:** When you enter the M command without specifying a value for *resource*, a list of the simulated devices currently being monitored is displayed. Coding other operands is not allowed.

**{ON,DISPLAY=** *name*|**OFF}**

**Function:** Activates or deactivates the Display Monitor Facility for the simulated device.

**Format:** You can specify the following values for this operand:

| | |
|---|---|
| **ON,DISPLAY=** *name* | Causes the Display Monitor Facility to go into session with the display specified by *name*. All Display Monitor Facility activity for *resource* is shown on this display. The display specified by *name* must be a 3270 display. |
| **OFF** | Causes the Display Monitor Facility to deactivate the session with the display monitoring the activity for *resource*. Coding other operands is not allowed. |

**AID={ON|OFF|(***row*,*column***)}**

**Function:** Specifies whether or not the attention identifier (AID) value generated by *resource* will be displayed on the monitoring display screen (*name*) when UPDATE=XMITRECV and VIEW=SCREEN are specified.

**Format:** When you specify ON, the AID value will be displayed on the monitoring display screen in the lower right corner. When you specify OFF, the AID value will not be displayed. When you specify (*row,column*), the AID value will be displayed on the monitoring display screen at the specified screen location, where *row* and *column* are integer values from 1 to 255. If the row and column values specified cause the AID display field to be displayed on more than one row or to be displayed outside of the monitor display screen, the AID display field will be moved to the nearest monitor display screen location.

**Default:** ON. This operand is optional.

**Note:** This operand is valid only when you specify VIEW=SCREEN and UPDATE=XMITRECV.

**CODE={EBCDIC|ASCII}**

**Function:** Specifies whether to display monitored data in EBCDIC or ASCII.

**Format:** EBCDIC or ASCII.

**Default:** EBCDIC

**DLOGMOD=** *logname*

**Function:** Specifies the name of the VTAM Logon Mode Table entry used for session initiation between *name* and the Display Monitor Facility. This operand is optional.

**LINES={***number***|2}**

**Function:** Specifies the number of lines shown each time the monitored device transmits or receives data.

**Format:** *number* can be an integer from 1 to the number of lines on the Display Monitor Facility display, up to a maximum of 99 lines. For SNA devices, the minimum number of lines used is two because one of the lines (*number*) specified is used for SNA formatting.

**Default:** 2. This operand is optional.

**Note:** This operand is valid only when you specify VIEW=DATA.

**SOURCE={ BLOCKS |DATA}**

**Function:** If the screen size of *resource* is larger than the screen size of*name*, specifying SOURCE=BLOCKS truncates the display image to fit the monitoring display. Specifying SOURCE=DATA does not truncate display images to fit the monitoring display. This may result in an error if the screen sizes are different.

**Format:** When you specify BLOCKS, the internal control blocks are used. When you specify DATA, the data stream transmitted or received by the simulated 3270 display is used.

**Default:** BLOCKS. This operand is optional.

**Notes:**

1. This operand is valid only when you specify VIEW=SCREEN and UPDATE=XMITRECV.
2. When you specify SOURCE=BLOCKS, the following occurs:

- Programmed symbols are not shown on the display monitor screen image. However, extended field and character attributes for color and highlighting are shown on the monitoring display.
- Double-byte character set (DBCS) data is displayed when both the simulated and monitor displays support DBCS and have the same screen sizes.
- Field outlining attributes are displayed when the monitor display supports field outlining.
- DBCS data is displayed on a non-DBCS monitor display as single-byte character set data with SO and SI characters shown as < and > respectively.

3. When you specify SOURCE=DATA, your display image is not updated to show the results of the following:
   - An erase input (ERIN) statement is issued by the simulated display.
   - An erase end of field (EREOF) statement is issued by the simulated display.
   - The location of data entered during the message generation process by the simulated display that is preceded by nulls that are suppressed when the message is sent to the host application. The location of the data on the monitor screen is shifted left and overlays the suppressed nulls.
   - Data streams rejected by the simulated display are not passed to the monitor display.
4. When you specify SOURCE=DATA and the simulated display has DBCS=YES specified in the network definition and the monitor display also supports 3270 DBCS data streams, the monitor display may reject data streams for one of the following reasons:
   - The host application program created a DBCS subfield with imbedded nulls that are suppressed if the DBCS subfield is sent back to the host application. In this case, an SI character is not paired with an SO character on the monitor display.
   - Suppressed nulls cause the data sent to the host application program to be shifted left and cause a DBCS character to be split on the monitor display.

If this problem occurs with SOURCE=DATA, use SOURCE=BLOCKS to avoid this situation. As a general rule, use SOURCE=BLOCKS when monitoring simulated 3270 DBCS displays.

**TIMER={ *integer*| <u>10</u> }**

**Function:** Specifies the timer value in seconds that is used when you specify UPDATE=TIMER.

**Format:** *integer* is from 1 to 600.

**Default:** 10 seconds. This operand is optional.

**Note:** This operand is valid only when you specify VIEW=SCREEN and UPDATE=TIMER.

**UPDATE={MONITOR|TIMER| <u>XMITRECV</u> }**

**Function:** Specifies when the monitor display is updated.

**Format:** When you specify MONITOR, the monitor display is updated each time a MONITOR statement is processed during message generation by the monitored display (*resource*). When you specify XMITRECV, the monitor display is updated as data is transmitted or received by the monitored display (*resource*). When you specify TIMER, the monitor display is updated each time the specified or defaulted timer value expires.

**Default:** XMITRECV. This operand is optional.

**Note:** This operand is valid only when you specify VIEW=SCREEN.

**VIEW={DATA|SCREEN}**

**Function:** Specifies whether screen images or data flows are to be shown on the monitor display.

**Format:** When you specify DATA, the monitored device's (*resource*) transmitted and received data flows are shown on the monitor display. When you specify SCREEN, the monitored device's (*resource*) screen image is shown on the monitor display. You can only specify VIEW=SCREEN for monitoring simulated 3270 displays.

**Default:** SCREEN for a simulated 3270 display and DATA for a simulated non-3270 device.

**Notes:**

- You may abbreviate keyword operands and applicable values. For example, all of the following character strings specify UPDATE=MONITOR:

```
UP=M
U=MO
UPDA=MONIT
UPDATE=M
```

- Care should be taken in using abbreviations. The character string used must be long enough to uniquely identify the desired operand. The character string D could mean either DISPLAY or DLOGMOD. If you code the character string D for an operand, it will default to DISPLAY.

***Examples:*** The following are some sample formats for the M command:

```
M LU327-1,ON,DISPLAY=VTAMLU1,VIEW=SCREEN
```

This command activates a session between the Display Monitor Facility and VTAMLU1 and monitors the first half-session of LU3270. The following defaults are accepted: SOURCE=BLOCKS, TIMER=10, UPDATE=XMITRECV, AID=ON.

```
M LU327-1,OFF
```

This command deactivates the Display Monitor Facility session that was monitoring device LU3270-1. This session was between VTAMLU1 and the Display Monitor Facility.

```
M NDS36,ON,DI=VTAMLU3,V=DA,L=1
```

This command activates a session between the Display Monitor Facility and VTAMLU3 and monitors the NDS30060 device. The Display Monitor Facility monitors the data stream for device NDS30060 and shows 10 lines of text for each transmitted and received data stream.

## O-Output data

```
O
```

This command closes and reallocates the SYSPRINT data set, if the SYSPRINT data set was directly allocated at WSim startup (see Chapter 14, “Running WSim,” on page 91 for information about starting WSim). The dynamically allocated SYSPRINT data set under MVS is closed and reallocated, which frees any previously printed data to be written by the MVS writer routines.

This command is ignored if the SYSPRINT data set has not been dynamically allocated.

## P-Stop network resources

```
P [resource]
```

This command stops activity for a network or an APPCLU, VTAMAPPL, or TCPIP within the network. All statistics and I/O activity are halted for the specified resources. Stopped resources may be restarted by entering an S (Start) operator command.

*resource*

**Function:** Specifies the resource to be stopped.

**Format:** You can specify the following values for *resource*:

| | |
|---|---|
| *ntwrk* | Specifies a single network for which all APPC LUs, VTAM applications, and TCP/IP connections are to be stopped. |
| *appclu* | Specifies an APPCLU to be stopped. |
| *vtamappl* | Specifies a VTAMAPPL to be stopped. |
| *tcpip* | Specifies a TCP/IP connection to be stopped. |

**Default:** If you do not specify a value for *resource*, all active networks are to be stopped.

## Q-Query network resources

```
Q [resource[,area[,length]]]
```

This command displays the current status of a network resource at the operator's console.

*resource*

**Function:** Specifies the resource to be queried.

**Format:** You can specify the following values for *resource*:

| | |
|---|---|
| *ntwrk* | Specifies the network to be displayed. |
| *appclu* | Specifies the APPCLU to be displayed. |
| *vtamappl* | Specifies the VTAMAPPL to be displayed. |
| *tcpip* | Specifies the TCP/IP connection to be displayed. |
| *dev* | Specifies the device to be displayed. |

| | |
|---|---|
| *lu*[- *num*] | Specifies the LU session to be displayed, where *-num* indicates which session is to be displayed. If you omit - *num*, the first session of the specified LU is displayed. *lu-** is also accepted which resolves to the last session number. |
| *tp*[- *num*] | Specifies the simulated CPI-C transaction program to be displayed, where *-num* indicates which instance is to be displayed. If you omit - *num*, the first instance of the specified TP is displayed. *tp-** is also accepted which resolves to the last instance number. |
| *appclu.tp*[- *num*] | Specifies the simulated CPI-C transaction program to be displayed, where *-num* indicates which instance is to be displayed. If you omit - *num*, the first instance of the specified TP is displayed. *appclu.tp-** is also accepted which resolves to the last instance number. |

**Default:** If you do not specify a value for *resource* or *area*, all initialized networks and their status are to be displayed using console messages ITP006I and ITP012I.

*area*

**Function:** Specifies the save area or user area to be queried. If you query an area, you must specify the associated resource.

**Format:** You can specify the following values for the *area* operand:

| | |
|---|---|
| **N**± *intege r* | Indicates the offset within the network user area to begin displaying data. ± *integer* is an optional integer between 0 and 32766. N+ *integer* indicates an offset from the beginning of the network user area. N- *integer* indicates an offset from the end of the network user area. If you do not specify ± *integer*, the user area is displayed as if you specified a value of +0. |
| **U**± *intege r* | Indicates the offset within the device user area to begin displaying data. ± *integer* is an optional integer between 0 and 32766. U+ *integer* indicates an offset from the beginning of the device user area. U- *integer* indicates an offset from the end of the device user area. If you do not specify ± *integer*, the user area is displayed as if you specified a value of +0. |
| **N** *s*± *integer* | Indicates the offset within the network save area *s* to begin displaying data. *s* is an integer between 1 and 4095. N *s+ integer* indicates an offset from the beginning of the network save area. N *s- integer* indicates an offset from the end of the network save area. ± *integer* is an optional integer between 0 and the length of the save area. If you do not specify ± *integer*, the save area is displayed as if you specified a value of +0. |
| *s*± *intege r* | Indicates the offset within the device save area *s* to begin displaying data. *s* is an integer between 1 and 4095. *s+ integer* indicates an offset from the beginning of the device save area. *s- integer* indicates an offset from the end of the device save area. ± *integer* is an optional integer between 0 and the length of the save area. If you do not specify ± *integer*, the save area is displayed as if you specified a value of +0. |

*length*

**Function:** Specifies the length of the data to be displayed.

**Format:** You can specify the following values for the *length* operand:

| | |
|---|---|
| *number* | Indicates the length of the area to be displayed starting at the point specified by *area*. If *number* is greater than the length of the data in the area starting at the point specified by *area*, all of the data from that point is displayed. |
| * | Indicates that the entire area from the point specified by *area* is to be displayed. |

**Default:** 256.

## R-Reset a network

```
R[ntwrk]
```

This command clears all terminal statistics counters shown on interval reports for a specified network and resets the network to its initial condition. The R (Reset) command also clears online response-time statistics (RSTATS) for network resources. You must stop a network before you can reset it.

*ntwrk*

**Function:** Specifies the stopped network to be reset.

**Default:** If you do not specify a value for *ntwrk*, all stopped networks are to be reset.

## S-Start network resources

```
S [resource]
```

This command causes WSim to start activity for a network or an APPCLU, VTAMAPPL, or TCPIP within a network.

*resource*

**Function:** Specifies the resource to be started.

**Format:** You can specify the following values for *resource*:

| | |
|---|---|
| *ntwrk* | Specifies a single network for which all APPC LUs, VTAM applications and TCP/IP connections are to be started. |
| *appclu* | Specifies an APPCLU to be started. |
| *vtamappl* | Specifies a VTAMAPPL to be started. |
| *tcpip* | Specifies a TCP/IP connection to be started. |

**Default:** If you do not specify a value for *resource*, all networks previously initialized are to be started.

## T-Dispatcher traces

```
T [{ALL |DSP}
```

This command dumps trace tables to the printer. The trace that is always active is the dispatcher trace facility. The output produced by this command is provided as a source of information to help with debugging WSim problems. This internal information should never be used as programming interface information.

`{ ALL |DSP}`

**Function:** Specifies the types of trace tables to be dumped to the SYSPRINT data set.

**Format:** You can specify the following values for this operand:

| | |
|---|---|
| **ALL** | Specifies that the dispatcher trace is to be dumped to the SYSPRINT data set. |
| **DSP** | Specifies that the dispatcher trace entries are to be dumped to the SYSPRINT data set. |

**Default:** ALL

## W-RSTATS query

```
W resource
```

This command returns the RSTATS (response statistics) information for the specified resource.

*resource*

**Function:** Specifies the resource for which RSTATS information is to be returned.

**Format:** You can specify the following values for *resource*

| | |
|---|---|
| *dev* | Specifies the device to be queried. |
| *lu*[- *num*] | Specifies the LU session to be queried, where *-num* indicates which session within the LU. If you omit - *num*, the first session within the LU is queried. *lu-** is also accepted which resolves to the last session number. |

**Note:** You can use the A (Alter) and the R (Reset) operator commands to reset RSTATS. For information about the format of the RSTATS messages, refer to *WSim Script Guide and Reference*.

## Z-Closedown

```
Z END
ZEND
```

This command causes an orderly shutdown of the WSim system and writes the message log buffers to the log data set. It causes the job to end.

## *-Comment

```
* [comment_data]
```

This command performs no operation. Its purpose is to write a console type record to the log data set containing the comment data from the command.

This command has no operands other than the comment data.

The total length of the command and its comment data cannot exceed 120 characters.

## $-Exit

```
$ ntwrk,[data]
```

This command passes user data to a network-level user exit specified by the NETEXIT or UCMDEXIT operand.

This command is part of a general-use programming interface which allows the customer to write programs that use the services of WSim.

For more information about the $ (Exit) operator command and about user exits, refer to *WSim User Exits* and *Creating WSim Scripts*.

*ntwrk*

**Function:** Specifies the name of the NTWRK statement for which the command is entered. You must enter the comma, even if you specify no data.

*data*

**Function:** Specifies the data to be passed to the user exit. WSim validates the network name and then passes the data to the NETEXIT or UCMDEXIT associated with the network without validation or translation.

**Format:** The total length of a $ (Exit) command cannot exceed 120 characters. The length of the data passed to the user exit must be less than or equal to 120 minus *n*, where *n* is the length of $ *ntwrk*.

**Notes:**

- The EBCDIC data entered is not folded to uppercase.
- You can enter literal text DBCS data.

# 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

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

| | | |
|---|---|---|
| CICS® | InfoWindow | IMS™ |
| MVS | MVS/ESA | MVS/SP |
| MVS/XA | OS/390® | RETAIN® |
| Series/1 | SP | System/36 |
| System/38 | System/370 | Systems Application Architecture® |
| VTAM | | |

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

# 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.

**American National Standard Code for Information Interchange (ASCII).**
The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters. (A)

**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.

**ASCII.**
American National Standard Code for Information Interchange.

**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.

**Availability Monitor (AVMON).**
A sample set of network definition statements and message generation decks provided with WSim which monitors the availability of host application subsystems.

**available.**
In VTAM, pertaining to a logical unit that is active, connected, enabled, and not at its session limit.

**AVMON.**
Availability Monitor.

## B

**basic sequential access method (BSAM).**
An access method for storing or retrieving

data blocks in a continuous sequence, using either a sequential access or a direct access device.

**BSAM.**
Basic sequential access method.

## C

**chain.**
A group of logically linked records, for example, an SNA message.

**Common Programming Interface for Communications (CPI-C).**
(1) In WSim, an application programming interface (API) used to perform program-to-program communications using LU type 6.2 communication protocols. (2) 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

**DASD.**
Direct access storage device.

**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.

**DASD.**
Direct access storage device.

**ddname.**
Data definition name.

**DFC.** Data flow control.

**direct access storage device (DASD).**
A device in which the access time is effectively independent of the location of the data. For example, a disk.

**Display Monitor Facility.**
A VTAM application program within WSim that displays simulated 3270 screen images on a monitor. It is used to monitor a WSim simulation dynamically, enabling a user to debug scripts and view interactions with host applications.

**double-byte character set (DBCS).**
A set of characters in which each character is represented by two bytes. Languages such as Japanese, Chinese, and Korean, which contain more symbols that can be represented by 256 code points, require double-byte character sets. Because each character requires two bytes, the typing, display, and printing of DBCS characters requires hardware and programs that support DBCS.

**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.

**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.

## 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.

**FDX.** Full duplex.

**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.

**FTP.** File transfer protocol.

**full duplex (FDX).**
Synonym for duplex.

## G

**generalized trace facility (GTF).**
An optional OS/VS service program that records significant system events, such as supervisor calls and start I/O operations, for the purpose of problem determination.

**GTF.** Generalized trace facility.

## 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.

**instance.**
A copy of a transaction program that is operating on a logical unit. If multiple instances are supported on the logical unit, multiple copies of the same transaction program can operate simultaneously.

**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.

**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)

**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 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 sessionsone with an SSCP and one with another LUand 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

**message format service (MFS).**
In IMS/VS, an editing facility that allows application programs to deal with simple logical messages instead of device-dependent data, thus simplifying the application development process.

**message generation.**
In WSim, the process of executing 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.

**MFS.** Message format service.

**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

**NC.** Network control.

**NCB.** Network control block.

**network.**
The set of statements defining an entire network, including both the network definition statements and the message generation source statements. Should not be confused with a packet switching network.

**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

**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)

**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.

**PTN.** Partition control block.

**PU.** Physical unit.

## Q

**QSAM.**
Queued sequential access method.

**queued sequential access method (QSAM).**
An extended version of the basic sequential access method (BSAM). When this method is used, a queue is formed of input data blocks that are awaiting processing or of output data blocks that have been processed and are awaiting transfer to auxiliary storage or to an output device.

## 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.

**RNR.** Receive not ready.

**RR.** Receive ready.

**RU.** Request unit or response unit.

## S

**SA.** Set attribute.

**SC.** Session Control.

**script.** The set of statements defining an entire network, including both the network definition statements and the message generation source statements.

**Script Generator Utility.**
A utility that enables WSim to convert data captured from a system into message generation scripts.

**secondary logical unit (SLU).**
In SNA, the logical unit (LU) that contains the secondary half-session for a particular LU-LU session. An LU may contain secondary and primary half-sessions for different active LU-LU sessions. Contrast with primary logical unit (PLU).

**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.

**single-domain network.**
In SNA, a network with one system services control point (SSCP). Contrast with multiple-domain network.

**SI.** Shift In. Used with DBCS. This is the X'0F' character that ends DBCS data.

**SLU.** Secondary logical unit.

**SMP.** System Modification Program.

**SMP/E.**
System Modification Program-Extended.

**SNA.** Systems Network Architecture.

**SNA backbone network.**
A collection of SNA nodes used as a packet switching network.

**SO.** Shift Out. Used with DBCS. This is the X'0E' character that begins DBCS data.

**SSP.** System Support Program.

**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.

**System Modification Program (SMP).**
An operating system component that facilitates the process of installing and servicing an MVS system.

**System Modification Program-Extended (SMP-E.).**
An IBM licensed program that facilitates the process of installing and servicing an MVS system.

**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

**TG.** Transmission group.

**TH.** Transmission header.

**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.

**transmission priority (TP).**
In SNA, a rank assigned to a path information unit (PIU) that determines its precedence for being selected by the transmission group control component of path control for forwarding to the next subarea node of the route used by the PIU.

**TSO.** Time sharing option.

## U

**UA.** Unnumbered acknowledgment.

**user table.**
In WSim, one or more text data entries contained in a table format which may be referenced for logic testing and message generation.

**UTI.** User time interval.

**Virtual Storage Access Method (VSAM).**
An access method for direct or sequential processing of fixed and variable-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry-sequence), or by relative-record number.

**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.

**VSAM.**
Virtual Storage Access Method.

**VTAM.**
Virtual Telecommunications Access Method.

**VTAMPARS II.**
VTAM Performance Analysis Reporting System II.

## W

**Workload Simulator (WSim).**
IBM program product to simulate terminals and networks. It enables the user to test system performance and evaluate network design.

**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.

**WTO.** Write-to-operator.

**WTOR.**
Write-to-operator-with-reply.

# 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 Test Manager User's Guide and Reference*, SC31-8949

*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

## Related publications

*MVS Installation and Tuning Guide*, GC28-0681

*OS/VS2/MVS System Programming Library: Job Management*, GC28-0627

*OS/VS2/MVS System Programming Library: TSO/E Customization*, SC28-1380

*VTAM Network Implementation Guide*, SC31-6404

*SNA Formats and Protocol Reference Manual: Architecture Logic*, SC30-3112

*Systems Network Architecture: Reference Summary*, GA27-3136

*Systems Application Architecture Common Programming Interface Communications Reference*, SC26-4399-06. (TPNS does not support CPI-C functions that have been added in later releases of this document.)

# Index

Printed in USA

SC31-8948-01