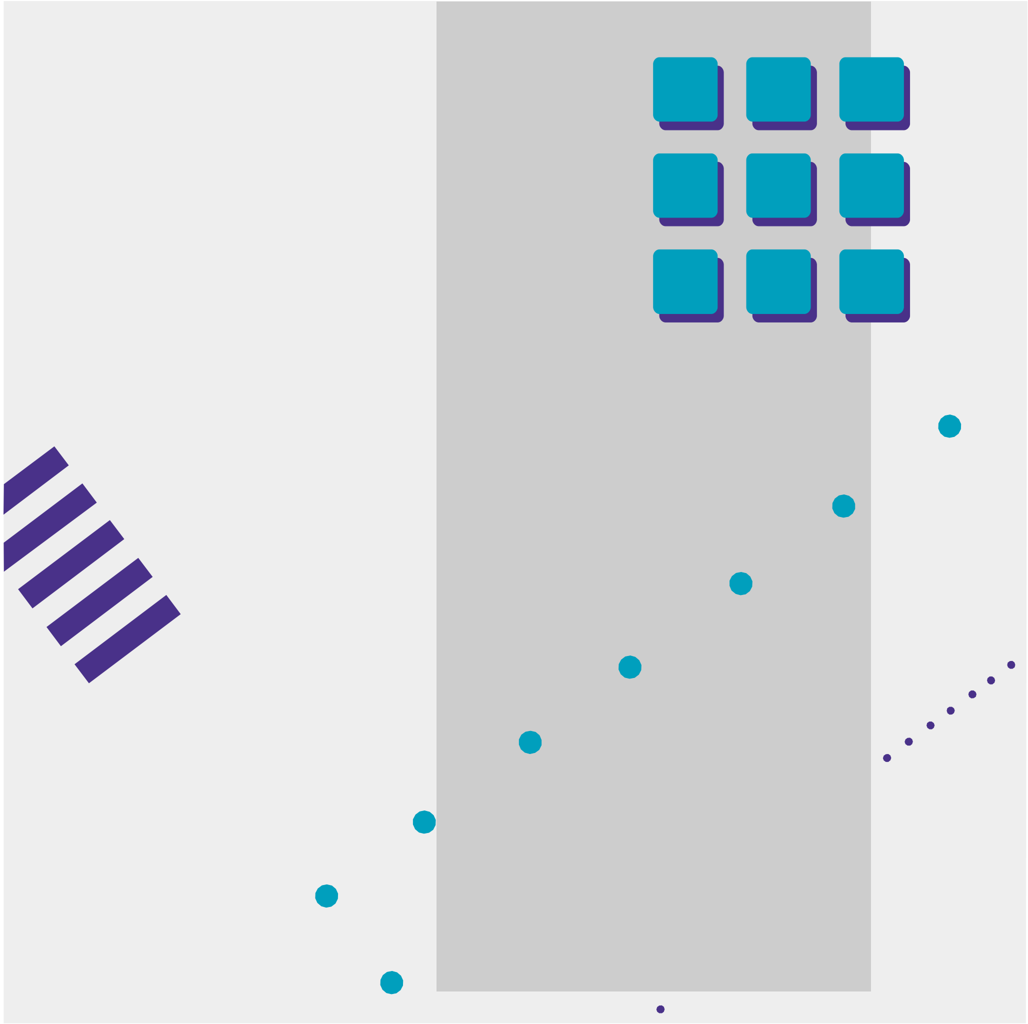
SC31-8950-00

WSim
Workload Simulator

User Exits

Version 1 Release 1



SC31-8950-00

WSim
Workload Simulator

User Exits

Version 1 Release 1

Note!

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

First Edition (August 2002)

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

MVS/370 (MVS/SP Version 1 or later)
MVS/Extended Architecture (MVS/SP Version 2 or later)
MVS/Enterprise System Architecture (MVS/SP Version 3 or later)
OS/390

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

Contents

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make them available in all countries in which IBM operates. 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 of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, New York 10594
United States of America

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement.

This document is not intended for production use and is furnished as is without any warranty of any kind, and all warranties are hereby disclaimed including the warranties of merchantability and fitness for a particular purpose.

Trademarks and Service Marks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

IMS
NetView
SP
MVS
OS/390
System/370
MVS/SP
Series/1
VTAM

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

About This Book

This book presents a discussion of how to write user exits for the Workload Simulator (WSim). It discusses the different types of user exits WSim provides and how to use them. The user exits for two WSim utility programs, the Loglist Utility and the Response Time Utility are also discussed.

Who Should Read This Book

Before you read this book, you should be familiar with information in *WSim Script Guide and Reference*. If you plan to write user exits for either the Loglist Utility or the Response Time Utility, you should also be familiar with the material applying to that utility in *WSim Utilities Guide*.

Read this book if you are responsible for coding WSim user exits. You should be familiar with the network concepts that are specific to the simulation for which you need to write the user exit.

How To Use This Book

Each chapter in this book presents a specific user exit interface. Once you have determined what function you want your user exit to perform, you should read the chapter that discusses the relevant user exit interface. Control block information is presented in the appendix.

This book contains the following chapters and appendixes:

- Chapter 1, "Coding Run Time User Exit Routines" on page 1 provides details on writing user exits that execute during a simulation run with WSim. It discusses the types of user exits available and how to invoke those exits. It also presents sample user exit programs.
- Chapter 2, "Coding Loglist Utility User Exit Routines" on page 41 discusses the user exits available for the Loglist Utility.
- Chapter 3, "Coding Response Time Utility User Exit Routines" on page 47 discusses the user exits available for the Response Time Utility.
- Appendix A, "Network-Level Exit (EXIT Operand)" on page 51 discusses the EXIT operand and how to use this operand.
- Appendix B, "Simulated Resource Type Codes" on page 53 lists the WSim code values for each simulated resource type.
- Appendix C, "Log Display Record Formats" on page 55 describes the format of the log display records.
- Appendix D, "User Exit Control Blocks" on page 57 lists control blocks for writing WSim user exits.

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 83.

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

Chapter 1. Coding Run Time User Exit Routines

WSim is designed to cover a wide range of terminal simulations and network configurations. Occasionally, you may run into testing situations that require unique actions or functions. In such cases, you can write exit routines to customize WSim to meet these special requirements. For example, you can write user exits to coordinate message generation among several different LUs.

Exit Routines

You can code an exit routine in the following situations:

- A simulated terminal receives input data (INEXIT).
- A simulated terminal prepares to send output data (OUTEXIT).
- A network is initialized, reset, or cancelled (NCTLEXIT).
- A $ (User Exit) operator command executes (UCMDEXIT).
- Any of the above situations occur and no specific exit is supplied (NETEXIT).
- An informational message is generated (INFOEXIT).
- An operator command issued from the user exit interface routine completes (UXOCEXIT).
- An EXIT that a message generation statement executes (EXIT).

Operands on the NTWRK statement define the exit routine called in each of the above situations.

Notes:

1. WSim also has the EXIT operand on the NTWRK statement. This operand defines an exit routine to be called on message input or output. This operand is obsolete and cannot be coded with INEXIT, OUTEXIT, UCMDEXIT, NCTLEXIT, or NETEXIT. It is recommended that you do not use this operand. See Appendix A, "Network-Level Exit (EXIT Operand)" on page 51 for more information on this operand.
2. When using the same exit routine in multiple networks, WSim loads only one copy of the exit, regardless of the attributes assigned by the linkage editor. It is this copy of the exit that is called for each invocation of the exit, no matter for what reason or network. If you plan to use user exits in this manner, you should take into account local storage, network user areas, and GETMAIN storage use.
3. When simulating CPI-C transaction programs (TPs), only the message generation exit may be utilized.

Register Linkage

When any type of exit routine is called, the registers are set as follows:

Register	**Usage**
1	Address of the parameter list
13	Address of a standard 18 fullword save area
14	Return address

15 Address of the module being called

The user exit must save the register contents and then restore them prior to returning to WSim. Also, the return code must be set in register 15 on return to WSim.

Return Codes

Return codes are only applicable for the Message Generation Exit (EXIT statement), the Informational Message-Level Exit (INFOEXIT operand), and the exit interface routine.

Addressing Mode Considerations

WSim user exits receive control in the addressing mode in which WSim is executing. This means all user exits should be written to execute in that addressing mode. This is particularly important if WSim is executing in 31-bit addressing mode and your exit interfaces with another program (for example, an access method) that does not support 31-bit addressing. See your system programmer if you do not know what addressing mode WSim will be using.

Changing Device Parameters

Of the fields defined in the device control block, the user exit routine must not modify the various addresses, the buffer length, the length of the user area or attribute table, or the session number.

You can reference and change the user switches, log byte, counters, AID byte, user area, and save areas. The output buffer, attribute or format table, cursor position, and attribute count can be modified according to the following information. Refer to Appendix D, "User Exit Control Blocks" on page 57 for a description of the control blocks you may need for WSim user exits.

Where possible, use the exit interface routine to obtain the addresses of the needed fields.

For non-display devices when you use the message generation EXIT statement, the cursor value (DEVCURSR) is an index to the next available byte in the buffer and is used to calculate message lengths. If the user exit places more data in the buffer, the cursor should be incremented beyond the data entered. For SNA, the SNA headers begin at the start of the buffer (DEVCURSR=1). The actual request/response unit (RU) is after these headers. When WSim enters message generation, the cursor is positioned at the first byte after the headers.

For display devices, the cursor value (DEVCURSR for 5250, PTNCCP for 3270) is the index (reference one) of the actual cursor that would appear on the screen.

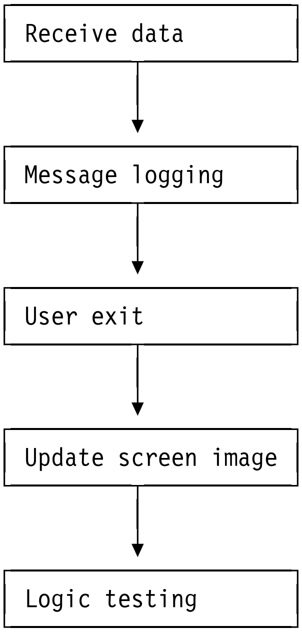
For 5250 displays, WSim maintains the screen image and format table as in the actual hardware. You must follow the functional specifications of the device when entering data on the screen or when modifying the screen or format table.

3101 displays, changing parameters for 3643 displays, changing parameters for 8775 displays, changing parameters for For the 3270 display, the actual screen image is maintained in the buffer, including attribute bytes. In addition, an attribute byte table is maintained in order to indicate which bytes in the buffer are attribute bytes. The table is one-eighth the size of the screen buffer, with each bit corresponding to a byte on the screen. When a bit in this table is set on, the corresponding byte on the screen is an attribute byte. A count of attribute bytes is also maintained.

If the user exit changes the number and/or position of the attribute bytes, the attribute table and count must also be changed. If data is being entered on the screen by the exit, you must find the attribute byte for the field and ensure that data can be entered in the field. Also, the exit must ensure that the modified data tag bit (MDT) is set when the data is entered. For 3270, the attribute byte format is the same as maintained in the hardware, except that bits 0 and 1 of the attribute byte are always zero.

Message Input Exit (INEXIT Operand)

By coding the INEXIT operand on the NTWRK statement, you indicate the name of an exit routine to be invoked after a simulated terminal receives data, but before the terminal updates its screen and performs any logic testing of the data. The sequence of events is shown below.



This exit could be used to examine and modify input to WSim, such as translating SDLC input from ASCII to EBCDIC.

Parameter List

Register 1 contains the address of the parameter list for the message input user exit specified by the INEXIT operand. The parameter list consists of the following addresses:

Word 1 The address of the data that was received. The data includes information received, except for line control characters, if applicable. For SNA devices, the data begins with TH, followed by the RH and RU fields.

Word 2 The address of a fullword containing the address of the WSim interface routine for the user exit. See "Exit Interface Routine" on page 30 for more information.

Word 3 The address of the device control block (DEV) or terminal control block (TRM) for the device or logical unit associated with the exit call. See Appendix D, "User Exit Control Blocks" on page 57 for more information on these control blocks.

Word 4 The address of a 16-bit field containing the following information:

- **Bit 0** Set ON (B'1') to indicate input to WSim.
- **Bit 1** Set ON (B'1') only by the user exit. If ON, processing for the current message stops. This bit is used for SNA terminal types only.
- **Bit 2** Set ON (B'1') only by the user exit. If ON, the current message generation delay will be cancelled.
- **Bits 3-4** Set OFF (B'0').
- **Bit 5** Set ON (B'1') to indicate INEXIT, OUTEXIT, NCTLEXIT, UCMDEXIT, or NETEXIT user exit.
- **Bit 6** Set ON (B'1') for normal input/output processing.
- **Bits 7-15** Set OFF (B'0').

Viewed another way, the contents of that 16-bit field when the exit routine is called are shown below:

```
1000 0110 0000 0000
```

Word 5 The address of the halfword containing the length of the data received. The exit program can be used to change this field, but it cannot be larger than the length of the terminal buffer (pointed to by Word 6) and must not be set to zero.

Word 6 The address of a halfword that contains the maximum length to which the data can be expanded. If you code INXEXPND=YES on the NTWRK statement, this is always the length of the terminal buffer (as specified by the BUFSIZE operand). If you code INEXPND=NO or omit this operand, this length is limited to the length of the input data for many terminal types. Do not change this field using the exit program.

Sample INEXIT Routine

The following sample message input exit program translates received data from 8-bit ASCII to EBCDIC.

```
         TITLE 'Sample INEXIT - Input Data Exit'
*---------------------------------------------------------------------*
* This routine is a sample input exit to translate received data      *
* from 8 bit ASCII to EBCDIC.  This exit performs this function       *
* for any SNA device.  It translates the RU portion of all received   *
* unformatted Function Management Data requests that have the         *
* alternate code bit set in the request/response header and it        *
* then resets that bit.  To locate the RU, it checks the TH FID       *
* type and increments past the TH and RH.                             *
*                                                                     *
* Use of this exit would be indicated by coding the INEXIT operand    *
* on the NTWRK statement (INEXIT=ASCIIIN).                            *
*---------------------------------------------------------------------*
ASCIIIN  CSECT
         STM   14,12,12(13)         Save caller's registers
         LR    12,15                Establish base register
         USING ASCIIIN,12
*
         LR    3,1                  Parameter list address
         L     4,8(3)               Get DEV address
         USING DEV,4                Use DEV DSECT
         TM    DEVTYPE,DEVSNA       Test for SNA device type
         BNO   RETURN               Skip translation if not SNA
         L     5,0(3)               Get address of data
         L     6,16(3)              Get length
         LH    6,0(6)                   of data
         LA    8,2                  Length of FID3 TH
         MVC   FIDTYPE,0(5)         Get byte containing FID type
         NI    FIDTYPE,X'F0'        Isolate FID type
         CLI   FIDTYPE,X'20'        Test for FID2
         BE    FID2                 Branch if found
         CLI   FIDTYPE,X'30'        Test for FID3
         BE    FIDFOUND             Branch if found
         CLI   FIDTYPE,X'40'        Test for FID4
         BNE   RETURN               Return if not FID4
         LA    8,20(8)              Add FID4/FID2 length difference
FID2     LA    8,4(8)               Add FID2/FID3 length difference
FIDFOUND LA    7,0(8,5)             Point to RH
         LA    8,3(,8)              Add length of RH
         LA    9,0(8,5)             Point to RU
         SLR   6,8                  Get length of RU
         BNO   RETURN               Branch if nothing to translate
         TM    0(7),X'E8'           Test for unformatted FM Data
         BNZ   RETURN                   request
         TM    2(7),X'08'           Test for alternate code
         BZ    RETURN               Not in alternate code
         NI    2(7),X'F7'           Reset alternate code
```

```
*
*  Translate the data from ASCII to EBCDIC
*
         LA    8,256                Translate 256 at a time
TRLOOP   CLR   6,8                  256 or less left
         BNH   TRREST               Go translate last if so
         TR    0(256,9),ASC2EBC     Translate 256 bytes
         SLR   6,8                  Decrement count
         ALR   9,8                  Increment data pointer
         B     TRLOOP
TRREST   BCTR  6,0                  Decrement for translate
         EX    6,TRRMDR             Translate remainder of data
*
RETURN   LM    14,12,12(13)         Restore registers
         BR    14                   Return to caller
*
TRRMDR   TR    0(0,9),ASC2EBC       Translate remainder of data
*
FIDTYPE  DS    C                    Workarea for determining FID type
*
ASC2EBC  DS    0CL256       An 8 bit ASCII to EBCDIC Translate Table
         DC    XL16'00010203372D2E2F1605250B0C0D0E0F'       0
         DC    XL16'101112133C3D322618193F27221D351F'       1
         DC    XL16'405A7F7B5B6C507D4D5D5C4E6B604B61'       2
         DC    XL16'F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F'       3
         DC    XL16'7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6'       4
         DC    XL16'D7D8D9E2E3E4E5E6E7E8E9ADE0BD5F6D'       5
         DC    XL16'79818283848586878889919293949596'       6
         DC    XL16'979899A2A3A4A5A6A7A8A9C04FD0A107'       7
         DC    XL16'4320211C23EB249B7128384990BAECDF'       8
         DC    XL16'45292A9D722B8A9A6756644A53685946'       9
         DC    XL16'EADA2CDE8B5541FE5851524869DB8E8D'       A
         DC    XL16'737475FA15B0B1B3B4B56AB7B8B9CCBC'       B
         DC    XL16'AB3E3B0ABF8F3A14A017CBCA1A1B9C04'       C
         DC    XL16'34EF1E0608097770BEBBAC5463656662'       D
         DC    XL16'30424757EE33B6E1CDED3644CECF31AA'       E
         DC    XL16'FC9EAE8CDDDC39FB80AFFD7876B29FFF'       F
*
*        Copy DSECTS from WSim User Exit Interface Library
*
         COPY  DEV
         END
```

ITPGSIPA Exit Routine

The exit routine which can be used to glean the address information from received data and set the address information for data to be transmitted is ITPGSIPA. This exit can be used as an input user exit to retrieve the full address of the source of the last data received for Simple UDP or Simple TCP simulated devices or as a message generation exit to set the full address to be used for the next message to be transmitted for SUDP devices or for the next connection established for STCP devices. When called as an input exit (specified by INEXIT on the NTWRK statement), ITPGSIPA saves the full INET address for the message being received by Simple TCP or Simple UDP terminals in network save area 13. The address is in the form used by the sockets interface, which is as follows:

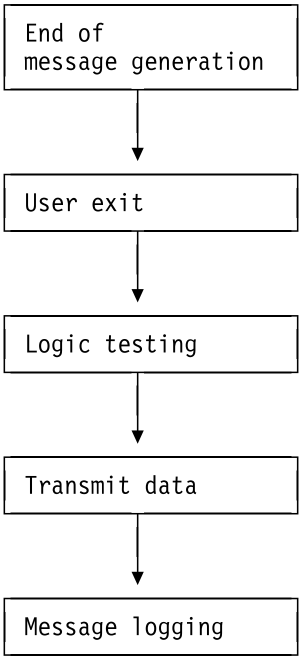
Table 1. INET Address Format

Offset	Length	Description
0	2	Address Family
2	2	Port(AF_NET=0002)
4	4	IP Address
8	8	Binary Zeros

When called as a Message Generation exit (USEREXIT STL statement or EXIT statement in WSim Scripting Language), ITPGSIPA moves the INET address from network save area 13 into the internal WSim control block so that the new address will be used for the next message transmitted (SUDP) or the next connection (STCP). ITPGSIPA assumes that the INET address exists in the save area in the format described in Table 1.

Message Output Exit (OUTEXIT Operand)

By coding the OUTEXIT operand on the NTWRK statement, you indicate the name of an exit routine to be invoked after a simulated terminal prepares to send data, but before the terminal performs any logic testing of the data. The sequence of events is shown below.



This exit could be used to examine and modify output from WSim, such as translating SDLC output from EBCDIC to ASCII.

Parameter List

Register 1 contains the address of the parameter list for the message output user exit specified by the OUTEXIT operand. The parameter list consists of the following addresses:

Word 1 The address of the data that will be sent. The data includes information to be sent, except for line control characters, if applicable. For SNA devices, the data begins with TH, followed by the RH and RU fields.

Word 2 The address of a fullword containing the address of the WSim interface routine for the user exit. See "Exit Interface Routine" on page 30 for more information.

Word 3 The address of the device control block (DEV) or terminal control block (TRM) for the terminal, device, or logical unit associated with the exit call. See Appendix D, "User Exit Control Blocks" on page 57 for more information on these control blocks.

Word 4 The address of a 16-bit field containing the following information:

Bit 0 Set OFF (B'0') to indicate output from WSim.

Bit 1 Set ON (B'1') only by the user exit. If ON, the message will not be sent or logged. This bit is used for SNA terminal types only.

Bit 2 Set ON (B'1') only by the user exit. If ON, the current message generation delay will be cancelled.

Bits 3-4 Set OFF (B'0').

Bit 5 Set ON (B'1') to indicate INEXIT, OUTEXIT, NCTLEXIT, UCMDEXIT, or NETEXIT user exit.

Bit 6 Set ON (B'1') for normal input/output processing.

Bits 7-15 Set OFF (B'0').

Viewed another way, the contents of that 16-bit field when the exit routine is called are shown below:

```
0000 0110 0000 0000
```

Word 5 The address of the halfword containing the length of the data sent. The exit program can be used to change this field, but it cannot be larger than the length of the terminal buffer (pointed to by Word 6) and must not be set to zero.

Word 6 The address of a halfword that contains the length of the terminal buffer, which is the maximum length to which the data can be expanded. Do not change this field using the exit program.

Sample OUTEXIT Routine

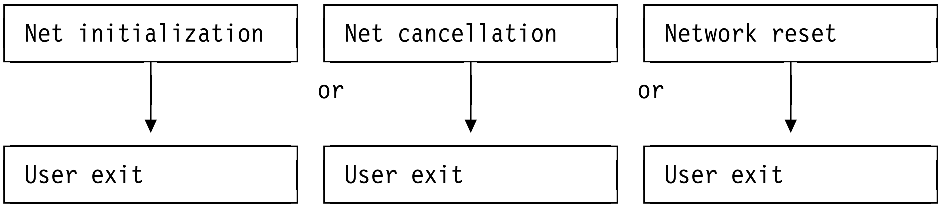
The following sample message output exit program translates generated data from EBCDIC to 8-bit ASCII.

```
         TITLE 'Sample OUTEXIT - Output Data Exit'
*----------------------------------------------------------------------*
* This routine is a sample output exit to translate generated data     *
* from EBCDIC to 8 bit ASCII.  This exit performs this function for    *
* any SNA device.  It translates the RU portion of all generated       *
* unformatted Function Management Data requests and sets the           *
* alternate code bit in the request/response header.                   *
* To locate the RU, it checks the TH FID type and increments past      *
* the TH and RH.                                                       *
*                                                                      *
* Use of this exit would be indicated by coding the OUTEXIT operand    *
* on the NTWRK statement (OUTEXIT=ASCIIOUT).                           *
*----------------------------------------------------------------------*
ASCIIOUT CSECT
         STM   14,12,12(13)         Save caller's registers
         LR    12,15                Establish base register
         USING ASCIIOUT,12

*
         LR    3,1                  Parameter list address
         L     4,8(3)               Get DEV address
         USING DEV,4                Use DEV DSECT
         TM    DEVTYPE,DEVSNA       Test for SNA device type
         BNO   RETURN               Skip translation if not SNA
         L     5,0(3)               Get address of data
         L     6,16(3)              Get length
         LH    6,0(6)                   of data
         LA    8,2                  Length of FID3 TH
         MVC   FIDTYPE,0(5)         Get byte containing FID type
         NI    FIDTYPE,X'F0'        Isolate FID type
         CLI   FIDTYPE,X'20'        Test for FID2
         BE    FID2                 Branch if found
         CLI   FIDTYPE,X'30'        Test for FID3
         BE    FIDFOUND             Branch if found
         CLI   FIDTYPE,X'40'        Test for FID4
         BNE   RETURN               Return if not FID4
         LA    8,20(8)              Add FID4/FID2 length difference
FID2     LA    8,4(8)               Add FID2/FID3 length difference
FIDFOUND LA    7,0(8,5)             Point to RH
         LA    8,3(,8)              Add length of RH
         LA    9,0(8,5)             Point to RU
         SLR   6,8                  Get length of RU
         BNO   RETURN               Branch if nothing to translate
         TM    0(7),X'E8'           Test for unformatted FM Data
         BNZ   RETURN                   request
         OI    2(7),X'08'           Set alternate code
```

```
*
*  Translate the data from EBCDIC to ASCII
*
         LA    8,256                Translate 256 at a time
TRLOOP   CLR   6,8                  256 or less left
         BNH   TRREST               Go translate last if so
         TR    0(256,9),EBC2ASC     Translate 256 bytes
         SLR   6,8                  Decrement count
         ALR   9,8                  Increment data pointer
         B     TRLOOP
TRREST   BCTR  6,0                  Decrement for translate
         EX    6,TRRMDR             Translate remainder of data
*
RETURN   LM    14,12,12(13)         Restore registers
         BR    14                   Return to caller
*
TRRMDR   TR    0(0,9),EBC2ASC       Translate remainder of data
*
FIDTYPE  DS    C                    Workarea for determining FID type
*
EBC2ASC  DS    0CL256       An EBCDIC to 8 bit ASCII translate table
         DC    XL16'00010203CF09D37FD4D5C30B0C0D0E0F'      0
         DC    XL16'10111213C7B408C91819CCCD831DD21F'      1
         DC    XL16'81821C84860A171B89919295A2050607'      2
         DC    XL16'E0EE16E5D01EEA048AF6C6C21415C11A'      3
         DC    XL16'20A6E180EB909FE2AB8B9B2E3C282B7C'      4
         DC    XL16'26A9AA9CDBA599E3A89E21242A293B5E'      5
         DC    XL16'2D2FDFDC9ADDDE989DACBA2C255F3E3F'      6
         DC    XL16'D78894B0B1B2FCD6FB603A2340273D22'      7
         DC    XL16'F861626364656667686996A4F3AFAEC5'      8
         DC    XL16'8C6A6B6C6D6E6F7071729787CE93F1FE'      9
         DC    XL16'C87E737475767778797AEFC0DA5BF2F9'      A
         DC    XL16'B5B6FDB7B8B9E6BBBCBD8DD9BF5DD8C4'      B
         DC    XL16'7B414243444546474849CBCABEE8ECED'      C
         DC    XL16'7D4A4B4C4D4E4F505152A1ADF5F4A38F'      D
         DC    XL16'5CE753545556575859AA0858EE9E4D1'       E
         DC    XL16'30313233343536373839B3F7F0FAA7FF'      F
*
*        Copy DSECTS from WSim User Exit Interface Library
*
         COPY  DEV
         END
```

Network Control Exit (NCTLEXIT Operand)

By coding the NCTLEXIT operand on the NTWRK statement, you indicate the name of an exit routine to be invoked at the end of network initialization, reset, or cancellation. The sequence of events is shown below.



This exit could be used to set up storage tables and control block data during network initialization.

Parameter List

Register 1 contains the address of the parameter list for the network control user exit specified by the NCTLEXIT operand. The parameter list consists of the following addresses:

Word 1 The address of a halfword of zeroes.

Word 2 The address of a fullword containing the address of the WSim interface routine for the user exit. See “Exit Interface Routine” on page 30 for more information.

Word 3 The address of a “dummy” device control block (DEV), whose only meaningful field will be DEVNCBAD, the address of the network control block (NCB) associated with the exit call. See “Device Control Block (DEV)” on page 57 for more information on this field.

Word 4 The address of a 16-bit field containing the following information:

- **Bits 0-4** Set OFF (B'0').
- **Bit 5** Set ON (B'1') to indicate INEXIT, OUTEXIT, NCTLEXIT, UCMDEXIT, or NETEXIT user exit.
- **Bit 6** Set OFF (B'0').
- **Bit 7** Set ON (B'1') for network initialization.
- **Bit 8** Set ON (B'1') for network cancellation.
- **Bit 9** Set ON (B'1') for network reset.
- **Bits 10-15** Set OFF (B'0').

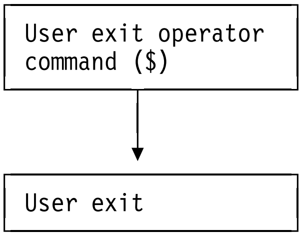
Viewed another way, the contents of that 16-bit field when the exit routine is called are shown below:

```
0000 0101 0000 0000 (initialization)
0000 0100 1000 0000 (cancellation)
0000 0100 0100 0000 (reset)
```

Words 5-6 The address of a halfword of zeroes.

Operator Command Exit (UCMDEXIT Operand)

By coding the UCMDEXIT operand on the NTWRK statement, you indicate the name of an exit routine to be invoked whenever an operator enters a $ (User Exit) command. The sequence of events is shown below.



This exit could be used to store data in a network save area.

Parameter List

Register 1 contains the address of the parameter list for the operator command user exit specified by the UCMDEXIT operand. The parameter list consists of the following addresses:

Word 1 The address of the start of the user parameter data that follows the comma after the network name. See *WSim User's Guide* for more information on the $ (User Exit) operator command.

Word 2 The address of a fullword containing the address of the WSim interface routine for the user exit. See “Exit Interface Routine” on page 30 for more information.

Word 3 The address of a “dummy” device control block (DEV), whose only meaningful field will be DEVNCBAD, the address of the network control block (NCB) associated with the exit call. See “Device Control Block (DEV)” on page 57 for more information on this field.

Word 4 The address of a 16-bit field containing the following information:

Bits 0-4 Set OFF (B'0').

Bit 5 Set ON (B'1') to indicate INEXIT, OUTEXIT, NCTLEXIT, UCMDEXIT, or NETEXIT user exit.

Bits 6-9 Set OFF (B'0').

Bit 10 Set ON (B'1') for $ (User Exit) operator command invocation.

Bits 10-15 Set OFF (B'0').

Viewed another way, the contents of that 16-bit field when the exit routine is called are shown below:

```
0000 0100 0010 0000
```

Words 5-6 The address of the halfword containing the length of the $ (User Exit) parameter data.

Sample UCMDEXIT Routine

The following sample operator command exit program takes the data entered by the $ (User Exit) operator command and places it into network savearea 255.

```
         TITLE 'OPEXIT - OPERATOR COMMAND EXIT ROUTINE'
*--------------------------------------------------------------------*
* This WSim operator command exit (UCMDEXIT=OPEXIT on WSim NTWRK      *
* statement) moves the data entered with a "$ network_name,data"      *
* operator command into network save area 255 for script reference.   *
* The WSim exit interface routine is used to allocate network save    *
* area 255.                                                            *
*--------------------------------------------------------------------*
OPEXIT   CSECT
         USING *,15                    USE R15 AS BASE REGISTER
         B     START                   BRANCH AROUND EYE CATCHER
         DC    CL8'OPEXIT  '           EYE CATCHER
         DROP  15                      RELEASE R15 AS BASE REGISTER
START    STM   14,12,12(13)            ENTRY
         LR    12,15                        .
         USING OPEXIT,12                     .
         ST    13,SAVEA+4                     .
         LA    14,SAVEA                        .
         ST    14,8(,13)                        .
         LR    13,14                             LINKAGE
         LR    3,1                     R3 = ADDRESS OF PARAMETER LIST
*                                           PASSED TO EXIT
```

```
*----------------------------------------------------------------------*
* ALLOCATE NETWORK SAVE AREA 255 USING WSim EXIT INTERFACE ROUTINE     *
*----------------------------------------------------------------------*
         MVI   SARQNUM,X'FF'           SET SAVE AREA NUMBER 255
         L     5,16(,3)                R5 = ADDRESS OF OPERATOR COMMAND
*                                           DATA LENGTH
         MVC   SARQSIZE(2),0(5)        SET SAVE AREA SIZE WANTED
         L     2,8(,3)                 R2 = ADDRESS OF DUMMY DEV CB
         ST    2,PLISTDEV              SET DEV CB ADDRESS IN PLIST
         L     5,4(,3)                 R5 = ADDRESS OF ADDRESS OF EXIT
*                                           INTERFACE ROUTINE
         L     15,0(,5)                R15 = ADDRESS OF EXIT INTERFACE
*                                            ROUTINE
         LA    1,PLIST                 R1 = ADDRESS OF PARAMETER LIST
         BALR  14,15                   CALL EXIT INTERFACE ROUTINE TO
*                                      ALLOCATE NETWORK SAVE AREA 255
         LTR   15,15                   NETWORK SAVE AREA ALLOCATED OK?
         BNZ   ERROR                   BRANCH NO

*----------------------------------------------------------------------*
* MOVE OPERATOR COMMAND ($) DATA INTO NETWORK SAVE AREA ALLOCATED      *
*----------------------------------------------------------------------*
         L     1,16(,3)                R1 = ADDRESS OF DATA LENGTH
         LH    2,0(,1)                 R2 = LENGTH OF DATA
         L     1,SADAPTR               R1 = ADDRESS OF SAVE AREA DATA
         L     4,0(,3)                 R4 = ADDRESS OF OP COMMAND DATA
         LR    5,2                     R5 = LENGTH OF DATA
         BCTR  5,0                     R5 = LENGTH OF DATA - 1
         EX    5,MOVEDATA              MOVE DATA INTO SAVE AREA
         L     1,SADALPTR              R1 = ADDRESS OF SAVE AREA DATA
*                                           LENGTH
         STH   2,0(,1)                 SET LENGTH OF DATA SAVED
         B     RETURN                  RETURN TO CALLER
ERROR    WTO 'ERROR ALLOCATING NETWORK SAVE AREA 255'
RETURN   SLR   15,15                   SET ZERO RETURN CODE
         L     13,4(,13)               EXIT
         L     14,12(,13)                  .
         LM    00,12,20(13)                 .
         BR    14                            LINKAGE
SAVEA    DS    18F                     REGISTER SAVE AREA
MOVEDATA MVC   0(0,1),0(4)             MOVE DATA INTO NET SAVE AREA 255
PLIST    DS    0F                      EXIT INTERFACE ROUTINE PLIST
PLISTDEV DS    A                       ADDRESS OF DEV CB
         DC    A(REQNETSA)             ADDRESS OF REQUEST CODE
         DC    A(SAPARMS)              ADDRESS OF RETURNED PARMS AREA
REQNETSA DC    X'57'                   REQUEST NETWORK SAVE AREA
SAPARMS  DS    0F                      REQUEST SAVE AREA PARAMETERS
SADAPTR  DS    0F                      ADDRESS OF SAVE AREA DATA
SARQNUM  DS    XL1                     REQUEST SAVE AREA NUMBER
SARQSIZE DS    XL2                     REQUEST SAVE AREA SIZE
         DS    XL1
SADALPTR DS    A                       ADDRESS OF SAVE AREA DATA LENGTH
SALENPTR DS    A                       ADDRESS OF SAVE AREA SIZE
         END
```

Network-Level Exit (NETEXIT Operand)

By coding the NETEXIT operand on the NTWRK statement, you indicate the name of a single exit routine to be invoked in all the same situations as INEXIT (message input), OUTEXIT (message output), NCTLEXIT (network control), and UCMDEXIT (user exit operator command). However, individual exits take precedence over the NETEXIT operand. For example, if you code NETEXIT=A and OUTEXIT=B on the NTWRK statement, exit A gets control for message input, network control, and user exit operator commands, while exit B gets control only for message output.

Parameter List

Register 1 contains the address of the parameter list for the network-level user exit specified by the NETEXIT operand. The parameter list consists of the following addresses:

Word 1 The same values as INEXIT, OUTEXIT, NCTLEXIT, or UCMDEXIT, whichever are appropriate.

Word 2 The address of a fullword containing the address of the WSim interface routine for the user exit. See “Exit Interface Routine” on page 30 for more information.

Words 3-6 The same values as INEXIT, OUTEXIT, NCTLEXIT, or UCMDEXIT, whichever are appropriate.

Sample NETEXIT Routine

The following sample program issues a WTO and a WSim S (Start) operator command at network initialization time. When the program is called for input/output processing of output messages, the terminal type byte is inserted into the twelfth byte of each outgoing message.

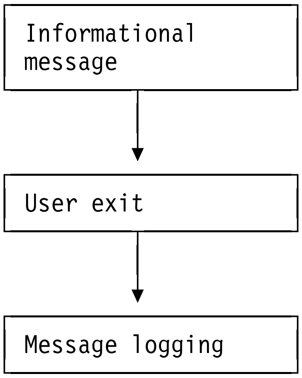
```
NETEXIT   CSECT                   NETWORK EXIT USING NETEXIT OPERAND
          STM   14,12,12(13)      SAVE CALLER'S REGISTERS
          BALR  12,0              SET UP BASE REGISTER
          USING *,12              TELL ASSEMBLER ABOUT IT
          ST    13,SAVEA+4        SAVE CALLER'S SAVE AREA ADDR
          LA    14,SAVEA          ADDR OF OUR SAVE AREA
          ST    14,8(13)          SAVE IN CALLER'S SAVE AREA
          LR    13,14             ADDR OF OUR SAVE AREA
          L     2,12(1)           GET ADDR OF CONTROL FLAGS
          TM    0(2),X'04'        NETEXIT CALL?
          BNO   RETURN            NO, IGNORE CALL
          TM    0(2),X'01'        CALLED FOR NETWORK INITIALIZATION?
          BNO   CKFORIOC          NO, CHECK FOR I/O CALL
```

```
*
*  NETEXIT CALLED AT NETWORK INITIALIZATION TIME
*  ISSUE WTO USING EXIT INTERFACE ROUTINE
*  ISSUE S (START) OPERATOR COMMAND USING EXIT INTERFACE ROUTINE
*
          MVC   INTPARM1(4),8(1)   ADDR OF DEV CB TO INTERFACE PARM1
          MVC   INTPARM2(4),8(1)   ADDR OF DEV CB TO INTERFACE PARM2
          L     15,4(1)            ADDR OF ADDR OF EXIT INTERFACE RTN
          L     15,0(15)           ADDR OF EXIT INTERFACE ROUTINE
          LR    3,15               SAVE INTERFACE ROUTINE ADDR IN REG 3
          LA    1,INTPARM1         ADDR OF INTERFACE PARAMETER LIST
          BALR  14,15              CALL EXIT INTERFACE TO ISSUE WTO
          LTR   15,15              FUNCTION COMPLETED OK?
          BNZ   RETURN             NO, QUIT
          LR    15,3               ADDR OF EXIT INTERFACE ROUTINE
          LA    1,INTPARM2         ADDR OF INTERFACE PARAMETER LIST
          BALR  14,15              CALL EXIT INTERFACE TO ISSUE
*                                  START (S) OPERATOR COMMAND
          B     RETURN             RETURN TO WSim
CKFORIOC  TM    0(2),X'02'         CALLED FOR INPUT/OUTPUT PROCESSING?
          BNO   RETURN             NO, IGNORE CALL
          TM    0(2),X'80'         OUTPUT MESSAGE?
          BO    RETURN             NO, IGNORE CALL

*
*  NETEXIT CALLED FOR OUTPUT PROCESSING
*  GET DEVICE TYPE BYTE USING EXIT INTERFACE ROUTINE
*  MOVE DEVICE TYPE BYTE INTO 12TH BYTE OF MESSAGE
*
          L     2,0(1)             GET ADDR OF DATA
          MVC   INTPARM3(4),8(1)   ADDR OF DEV CB TO INTERFACE PARM
          L     15,4(1)            ADDR OF ADDR OF EXIT INTERFACE RTN
          L     15,0(15)           ADDR OF EXIT INTERFACE ROUTINE
          LA    1,INTPARM3         ADDR OF INTERFACE PARAMETER LIST
          BALR  14,15              CALL EXIT INTERFACE TO GET TERM TYPE
          LTR   15,15              REQUEST COMPLETED OK?
          BNZ   RETURN             NO, QUIT
          L     11,DTYPEPTR        GET ADDR OF TERMINAL/DEVICE TYPE
          MVC   11(1,2),0(11)      MOVE TERMINAL/DEVICE CODE INTO MSG
RETURN    L     13,SAVEA+4         ADDR OF CALLER'S SAVE AREA
          LM    14,12,12(13)       RESTORE CALLER'S REGISTERS
          BR    14                 RETURN TO WSim
```

```
SAVEA     DS    18F                SAVE AREA
INTPARM1  DS    0F                 PARM LIST TO ISSUE WTO
          DS    A                  ADDR OF DEV CONTROL BLOCK
          DC    A(REQWTO)          ADDR OF REQUEST BYTE
          DC    A(WTOPARM)         ADDR OF DATA PARAMETER
WTOPARM   DC    A(WTODATA)         ADDR OF WTO DATA
          DC    AL2(L'WTODATA)     LENGTH OF WTO DATA
WTODATA   DC    C'** NETEXIT CALLED AT NETWORK INITIALIZATION TIME **'
REQWTO    DC    X'06'              REQUEST WTO FUNCTION
INTPARM2  DS    0F                 PARM LIST TO ISSUE OPERATOR COMMAND
          DS    A                  ADDR OF DEV CONTROL BLOCK
          DC    A(REQOPCMD)        ADDR OF REQUEST BYTE
          DC    A(OPCPARM)         ADDR OF DATA PARAMETER
OPCPARM   DC    A(OPCDATA)         ADDR OF OPERATOR COMMAND DATA
          DC    AL2(L'OPCDATA)     LENGTH OF OPERATOR COMMAND DATA
OPCDATA   DC    C'S'               WSim START (S) OPERATOR COMMAND
REQOPCMD  DC    X'05'              REQUEST OPERATOR COMMAND FUNCTION
INTPARM3  DS    0F                 PARM LIST TO PICK UP DEVICE TYPE
          DS    A                  ADDR OF DEV CONTROL BLOCK
          DC    A(REQDTYPE)        ADDR OF REQUEST BYTE
          DC    A(DTYPEPTR)        ADDR OF DATA PARAMETER
DTYPEPTR  DS    A                  ADDR OF DEVICE TYPE BYTE
REQDTYPE  DC    X'B6'              REQUEST DEVICE TYPE
          END
```

Informational Message-Level Exit (INFOEXIT Operand)

By coding the INFOEXIT operand on the NTWRK statement, you indicate the name of an exit routine to be invoked whenever WSim generates an informational message (ITP400 series messages) but before it writes the message to the log data set. The sequence of events is shown below.



This exit could be used to display informational message data at the WSim system console.

Parameter List

Register 1 contains the address of the parameter list for the informational message user exit specified by the INFOEXIT operand. The parameter list consists of the following addresses:

Word 1 The address of the informational message data.

Word 2 The address of a fullword containing the address of the WSim interface routine for the user exit. See "Exit Interface Routine" on page 30 for more information.

Word 3 The address of the device control block (DEV) for the device or logical unit associated with the exit call. See "Device Control Block (DEV)" on page 57 for more information on the DEV control block.

Word 4 The address of a 16-bit field containing the following information:

Bits 0-2 Set OFF (B'0').

Bit 3 Set ON (B'1') for informational message exit (INFOEXIT) call.

Bits 4-15 Set OFF (B'0').

Viewed another way, the contents of that 16-bit field when the exit routine is called are shown below:

```
0001 0000 0000 0000
```

Word 5 The address of the halfword containing the length of the informational message data. The exit program can be used to change this field, but it cannot be larger than the length of the informational message buffer (pointed to by Word 6) and must not be set to zero.

Word 6 The address of a halfword that contains the length of the informational message buffer, which is 152 bytes. Do not change this field using the exit program.

Return Codes

A non-zero return code passed back from the informational message exit in register 15 inhibits the informational message from being written to the WSim log data set.

Sample INFOEXIT Routine

The following sample informational message exit program takes the informational message data and displays it at the WSim system console using the exit interface routine WTO function.

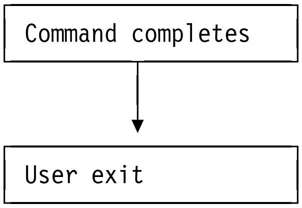
```
INFOEXIT CSECT                      INFO EXIT USING INFOEXIT OPERAND
         STM   14,12,12(13)         SAVE REGISTERS
         BALR  12,0                 SET UP BASE REGISTER
         USING *,12                 ESTABLISH ADDRESSABILITY
         ST    13,SAVEA+4           SAVE CALLER'S SAVE AREA ADDR
         LA    14,SAVEA             ADDR OF OUR SAVE AREA
         ST    14,8(13)             SAVE IN CALLER'S SAVE AREA
         LR    13,14                SAVE AREA ADDR
```

```
*
* IS EXIT CALLED AS INFORMATIONAL MESSAGE EXIT?
*
         L     RWORK,12(RPARM)     ADDR OF INPUT FLAGS
         TM    0(RWORK),IEXIT      INFORMATIONAL EXIT?
         BZ    GETOUT              NO, RETURN TO WSim

*
* BUILD EXIT INTERFACE ROUTINE PARAMETER LIST
*
         L     RWORK,8(RPARM)      ADDR OF DEV CONTROL BLOCK
         ST    RWORK,OUTPARM       PUT IN INTERFACE PARM LIST
         LA    RWORK,REQWTO        ADDR OF WTO REQUEST BYTE
         ST    RWORK,OUTPARM+4     PUT IN INTERFACE PARM LIST
         LA    RWORK,DATAPARM      ADDR OF DATA PARAMETER
         ST    RWORK,OUTPARM+8     PUT IT IN INTERFACE PARM LIST

*
* BUILD DATA PARAMETER
*
         L     RWORK,0(RPARM)      ADDR OF INFORMATIONAL MESSAGE DATA
         ST    RWORK,DATAPARM      PUT IN FIRST PART OF DATA PARM
         L     RWORK,16(RPARM)     ADDR OF DATA LENGTH
         LH    RWORK,0(RWORK)      LENGTH OF DATA
         STH   RWORK,DATAPARM+4    PUT IN INTERFACE PARM LIST

*
* CALL THE EXIT INTERFACE ROUTINE TO ISSUE THE WTO
*
         L     RPARMIN,4(RPARM)    ADDR OF EXIT INTERFACE ADDR
         L     RPARMIN,0(RPARMIN)  ADDR OF EXIT INTERFACE
         LA    RPARM,OUTPARM       ADDR OF OUTPUT PARM LIST
         LR    15,RPARMIN          ADDR OF EXIT INTERFACE ROUTINE
         BALR  14,15               CALL EXIT INTERFACE ROUTINE
         LTR   RCODE,RCODE         TEST RETURN CODE
         BZ    GETOUT              ZERO RETURN CODE

*
* IF EXIT INTERFACE HAD TROUBLE, ISSUE "BAD WTO" MESSAGE
* NOTE: PARAMETER LIST STILL SETUP TO ISSUE WTO
*
TROUBLE  EQU   *
         LA    RWORK,BADWTO        ADDR OF BAD WTO MESSAGE
         ST    RWORK,DATAPARM      PUT IN DATA PARAMETER
         LH    RWORK,BADWTOLN      LENGTH OF MESSAGE
         STH   RWORK,DATAPARM+4    PUT IN DATA PARAMETER
```

```
*
* CALL THE EXIT INTERFACE ROUTINE
*
         LA    RPARM,OUTPARM       ADDR OF OUTPUT PARM LIST
         LR    15,RPARMIN          ADDR OF EXIT INTERFACE ROUTINE
         BALR  14,15               CALL EXIT INTERFACE ROUTINE
GETOUT   EQU   *
         SR    RCODE,RCODE         SET RETURN CODE TO
*                                  WRITE INFO MSG TO LOG D/S
         L     13,SAVEA+4          CALLER'S SAVE AREA ADDR
         L     14,12(13)           RESTORE RETURN ADDRESS
         LM    0,12,20(13)         RESTORE OTHER REGISTERS
         BR    14                  RETURN TO WSim
*
SAVEA    DS    18F                 SAVE AREA
OUTPARM  DS    0F                  PARM LIST FOR EXIT INTERFACE
         DS    F                   ADDR OF DEV CONTROL BLOCK
         DS    F                   ADDR OF REQUEST BYTE
         DS    F                   ADDR OF DATA PARAMETER
DATAPARM DS    0F                  DATA PARAMETER
         DS    F                   ADDR OF DATA
         DS    H                   LENGTH OF DATA
BADWTO   DC    CL30'INTERFACE UNABLE TO ISSUE WTO '
BADWTOLN DC    H'30'               MESSAGE LENGTH
REQWTO   DC    X'06'               WTO REQUEST
IEXIT    EQU   X'10'               CALLED AS INFORMATIONAL EXIT
RPARM    EQU   1                   ADDR OF PARAMETERS
RDEV     EQU   2                   ADDR OF DEV CONTROL BLOCK
RWORK    EQU   3                   WORK REGISTER
RPARMIN  EQU   4                   SAVED ADDRESS OF INPUT PARMS
RCODE    EQU   15                  RETURN CODE
         END
```

User Exit Interface Command Exit (UXOCEXIT Operand)

By coding the UXOCEXIT operand on the NTWRK statement, you indicate the name of an exit routine to be invoked when an operator command issued by way of the User Exit Interface Routine (see “Exit Interface Routine” on page 30) by another exit using the X'07' request code completes. The sequence of events is shown below.



This exit can be used to issue operator commands from other exit routines and to notify those routines when the command finishes execution.

Parameter List

Register 1 contains the address of the parameter list for the line error user exit specified by the UOXCEXIT operand. The parameter list consists of the following addresses:

Word 1 The address of user data, which is the same as passed to the originally called exit routine. Typically, this is the address of a control block or save area.

Word 2 The address of a fullword containing the address of the WSim interface routine for the user exit. See "Exit Interface Routine" on page 30 for more information.

Word 3 The address of the device control block (DEV) for the device or logical unit associated with the exit call at the time the command was issued. See Appendix D, "User Exit Control Blocks" on page 57 for more information on these control blocks.

Word 4 The address of a 16-bit field containing the following information:

- **Bits 0-10** Set OFF (B'0').
- **Bit 11** Set ON (B'1') for user exit interface command exit (UXOCEXIT) call.
- **Bits 12-15** Set OFF (B'0').

Viewed another way, the contents of that 16-bit field are shown below:

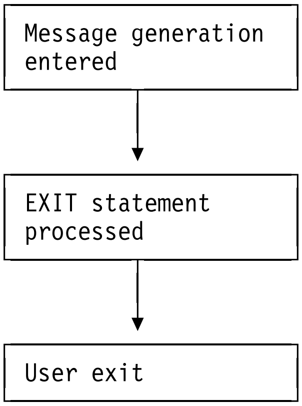
```
0000 0000 0001 0000
```

Words 5-6 Zero.

Message Generation Exit (EXIT Statement)

By coding the EXIT statement in a message deck, you indicate the name of an exit routine to be invoked whenever this statement is processed.

The sequence of events is shown below.



This exit could be used to access data that otherwise cannot be accessed by message generation decks.

Parameter List

Register 1 contains the address of the parameter list for the message generation user exit specified by the EXIT statement. The parameter list consists of the following addresses:

Word 1 The address of the user parameter (PARM operand data) coded on the EXIT statement. The first two bytes contain the length of the data coded in the parameter. The actual data directly follows the length. If the PARM operand is not specified, the length field will contain zeroes.

Word 2 The address of a fullword containing the address of the WSim interface routine for the user exit. See "Exit Interface Routine" on page 30 for more information.

Word 3 The address of the device control block (DEV) for the device, logical unit, or transaction program associated with the exit call. See "Device Control Block (DEV)" on page 57 for more information on the DEV control block. It is recommended that you use the WSim exit interface routine to obtain the addresses of fields in the DEV control block before referencing them.

Word 4 The address of a 16-bit field containing the following information:

Bits 0-3 Set OFF (B'0').

Bit 4 Set ON (B'1') for message generation exit (EXIT) call.

Bits 5-15 Set OFF (B'0').

Viewed another way, the contents of that 16-bit field when the exit routine is called are shown below:

```
0000 1000 0000 0000
```

Return Codes

When the user exit routine returns control to WSim message generation, the value of the return code in register 15 indicates the action taken by the exit routine. The following are valid return codes:

Code	Action
0	Continue in message generation as if the user exit had not been called.
4	Continue in message generation as though a delimiter had not been previously processed. For 3270 and 5250 terminals, a null AID is set.
8	A message was generated by the user exit routine. Continue processing as if the message had been generated by a TEXT statement.
12	Set the Wait indicator for the device and stop message generation processing. If a message was generated prior to calling the exit routine, transmit it now.
16	Stop message generation processing at this point, and do not set the Wait indicator for the device. If a message was generated prior to calling the exit routine, transmit it now.

Notes:

1. If any other code is returned, processing will continue as if the code returned were zero.
2. Return codes 0 and 4 result in the same action for STL programs, unless you include WSim Scripting Language statements in your STL program using @GENERATE. Refer to *WSim General Information* for more information about user exit considerations in STL.
3. When using this exit with CPI-C transaction program (TP) simulations, the save areas, counters, and user areas should be used as an interface with message generation for data passing. You can use the exit interface routine to gain access to these WSim resources.

Sample EXIT Statement Routine

The following sample program and message generation deck show how you could use a message generation exit routine instead of a TEXT statement to generate messages for a terminal. The sample program selects messages from a table according to a two-digit EBCDIC index value (01-10) passed in the EXIT statement parameter field. The exit can obtain the messages in another manner, such as reading from a data set. The sample message generation deck that invokes the exit routine assumes that the system under test will return a response resetting the terminal Wait indicator.

```
GENEXIT  CSECT                     MSG GEN EXIT USING EXIT STATEMENT
         STM   14,12,12(13)        SAVE REGISTERS
         BALR  12,0                SET UP BASE REGISTER
         USING *,12                ESTABLISH ADDRESSABILITY
         ST    13,SAVEA+4          SAVE CALLER'S SAVE AREA ADDR
         LA    14,SAVEA            ADDR OF OUR SAVE AREA
         ST    14,8(13)            SAVE IN CALLER'S SAVE AREA
         LR    13,14               SAVE AREA ADDR
         SR    RWORK,RWORK         SET DEFAULT RETURN CODE VALUE TO
         ST    RWORK,RETNCODE      INDICATE NO MESSAGE GENERATED

*
* WAS EXIT CALLED AS A MESSAGE GENERATION EXIT?
*
         L     RWORK,12(RPARM)     ADDR OF INPUT FLAGS
         TM    0(RWORK),MESSEXIT   MESSAGE DECK EXIT?
         BZ    GETOUT              NO, RETURN TO WSim

*
* GET INPUT PARAMETERS
*
         L     RPARMX,0(RPARM)     ADDR OF EXIT STMT PARM FIELD
         L     RPARMIN,4(RPARM)    ADDR OF USER INTERFACE ADDR
         L     RPARMIN,0(RPARMIN)  ADDR OF USER INTERFACE
         L     RDEV,8(RPARM)       ADDR OF DEV CONTROL BLOCK
         LH    RWORK,0(RPARMX)     LENGTH OF PARM DATA
         CH    RWORK,MINLENG       ENOUGH DATA ?
         BL    GETOUT              NO, RETURN TO WSim
```

```
*
* GET ADDRESS OF DEVICE DISPLAY (OUTPUT) BUFFER
* USING EXIT INTERFACE ROUTINE
*
         ST    RDEV,OUTPARM        PUT DEV ADDR IN INTERFACE PARM LIST
         LA    RWORK,REQDEVBF      ADDR OF DISPLAY BUFFER REQUEST BYTE
         ST    RWORK,OUTPARM+4     PUT IN INTERFACE PARM LIST
         LA    RWORK,RETAREA       ADDR OF INFORMATION RETURN AREA
         ST    RWORK,OUTPARM+8     PUT IN INTERFACE PARM LIST
         LA    RPARM,OUTPARM       ADDR OF OUTPUT PARM LIST
         LR    15,RPARMIN          ADDR OF EXIT INTERFACE ROUTINE
         BALR  14,15               CALL EXIT INTERFACE ROUTINE
         LTR   15,15               TEST RETURN CODE
         BNZ   GETOUT              NON-ZERO RETURN CODE
         L     RWORK,RETAREA       ADDRESS OF BUFFER
         ST    RWORK,BUFFADDR      SAVE BUFFER ADDRESS

*
* GET CURRENT CURSOR LOCATION (BUFFER INDEX) USING INTERFACE ROUTINE
*
         LA    RWORK,REQDEVCL      CURSOR OR BUFFER INDEX REQUEST
         ST    RWORK,OUTPARM+4     PUT IN INTERFACE PARM LIST
         LA    RPARM,OUTPARM       ADDR OF OUTPUT PARM LIST
         LR    15,RPARMIN          ADDR OF EXIT INTERFACE ROUTINE
         BALR  14,15               CALL EXIT INTERFACE ROUTINE
         LTR   15,15               TEST RETURN CODE
         BNZ   GETOUT              NON-ZERO RETURN CODE
         L     RWORK,RETAREA       ADDRESS OF CURSOR LOCATION
         ST    RWORK,BUFNDXAD      SAVE ADDR OF CURSOR LOCATION
         LH    RWORK,0(,RWORK)     CURSOR LOCATION
         STH   RWORK,BUFFNDEX      SAVE CURSOR LOCATION
```

```
*
*  WE HAVE BUFFER INFORMATION, NOW GENERATE MESSAGES
*
         LA    RTABLE,INDXTBL      ADDR OF INDEX TABLE
         LA    RINDEX,0            INITIALIZE INDEX
TSTINDEX EQU   *
         CLC   0(2,RWORK),2(RPARMX) FOUND INDEX ?
         BE    GENMSG              YES, GENERATE MESSAGE
         LA    RINDEX,1(RINDEX)    INCREMENT INDEX
         LA    RTABLE,2(RTABLE)    POINT TO NEXT TABLE ENTRY
         B     TSTINDEX            CHECK NEXT ENTRY
GENMSG   EQU   *
         M     RMULT,MSGLENG       OFFSET INTO MESSAGE TABLE
         LA    RMSG,MSGTBL         ADDR OF MESSAGE TABLE
         AR    RMSG,RINDEX         ADD OFFSET
         LH    RBUFF,BUFFNDEX      GET CURRENT BUFFER INDEX
         A     RBUFF,BUFFADDR      ADD ADDR OF OUTPUT BUFFER
         BCTR  RBUFF,0             SUBTRACT ONE
         L     RWORK,MSGLENG       GET MESSAGE LENGTH
         BCTR  RWORK,0             SUBTRACT ONE FOR MVC
         EX    RWORK,MSGMOVE       MOVE MESSAGE TO BUFFER
         LA    RWORK,1(RWORK)      TOTAL LENGTH
         AH    RWORK,BUFFNDEX      TOTAL DATA LENGTH
         STH   RWORK,BUFFNDEX      SET DEVICE DATA INDEX
         L     RWORK,BUFNDXAD      LOAD ADDRESS OF BUFFER INDEX
         MVC   0(2,RWORK),BUFFNDEX SET DATA INDEX IN DEV CONTROL BLOCK
         LA    RWORK,8             INDICATE MESSAGE GENERATED
         ST    RWORK,RETNCODE      IN RETURN CODE TO WSim
GETOUT   EQU   *
         L     RCODE,RETNCODE      SET RETURN CODE
         L     13,SAVEA+4          CALLER'S SAVE AREA ADDR
         L     14,12(13)           RESTORE RETURN ADDRESS
         LM    0,12,20(13)         RESTORE OTHER REGISTERS
         BR    14                  RETURN TO WSim
```

```
*
SAVEA    DS   18F                 SAVE AREA
MSGMOVE  MVC  0(0,RBUFF),0(RMSG)  MOVE MESSAGE TO BUFFER
MINLENG  DC   H'2'                MINIMUM LENGTH OF PARM FIELD
MSGLENG  DC   F'20'               LENGTH OF A MESSAGE
TBLEND   DC   C'00'               END OF INDEX TABLE
INDXTBL  DC   C'0102030405060708091000' INDEX TABLE
MSGTBL   DC   CL20'MESSAGE 1  '   MESSAGE TABLE
         DC   CL20'MESSAGE 2  '
         DC   CL20'MESSAGE 3  '
         DC   CL20'MESSAGE 4  '
         DC   CL20'MESSAGE 5  '
         DC   CL20'MESSAGE 6  '
         DC   CL20'MESSAGE 7  '
         DC   CL20'MESSAGE 8  '
         DC   CL20'MESSAGE 9  '
         DC   CL20'MESSAGE 10 '
MESSEXIT EQU  X'08'               CALLED AS MESSAGE DECK EXIT
REQDEVBF DC   X'B7'               DEVICE BUFFER REQUEST
REQDEVCL DC   X'BA'               DEVICE CURSOR (INDEX) REQUEST
OUTPARM  DS   0F                  PARM LIST FOR EXIT INTERFACE
         DS   F                   ADDR OF DEV CONTROL BLOCK
         DS   F                   ADDR OF REQUEST BYTE
         DS   F                   ADDR OF RETURN AREA
RETAREA  DS   2F                  INFORMATION RETURN AREA
BUFFADDR DS   F                   ADDR OF DEVICE OUTPUT BUFFER
BUFNDXAD DS   F                   ADDR OF DEVICE OUTPUT BUFFER INDEX
BUFFNDEX DS   H                   DEVICE BUFFER INDEX
RETNCODE DS   F                   RETURN CODE VALUE TO SET
RPARM    EQU  1                   ADDR OF PARAMETERS
RPARMX   EQU  2                   ADDR OF EXIT STMT PARM FIELD
RMSG     EQU  2                   ADDR OF MESSAGE TO GENERATE
RDEV     EQU  3                   ADDR OF DEV CONTROL BLOCK
RWORK    EQU  4                   WORK REGISTER
RTABLE   EQU  4                   ADDR OF INDEX TABLE
RMULT    EQU  6                   EVEN REGISTER FOR MULTIPLY
RBUFF    EQU  6                   ADDR OF OUTPUT BUFFER
RINDEX   EQU  7                   INDEX INTO MESSAGE TABLE
RPARMIN  EQU  8                   SAVED ADDRESS OF INPUT PARMS
RCODE    EQU  15                  RETURN CODE REGISTER
         END
```

The following are sample message generation EXIT statements.

```
DECK1    MSGTXT
1        IF   LOC=B+0,TEXT=('FF'),WHEN=IN,ELSE=CONT,STATUS=HOLD
         EREOF                         ERASE DATA IN FIELD AND SET
*                                      MODIFIED DATA TAG (MDT) BIT
         EXIT MODULE=GENEXIT,PARM=(01) INSERT DATA INTO FIELD
         WAIT
         EREOF                         ERASE DATA IN FIELD AND SET
*                                      MODIFIED DATA TAG (MDT) BIT
         EXIT MODULE=GENEXIT,PARM=(02) INSERT DATA INTO FIELD
         WAIT
         EREOF                         ERASE DATA IN FIELD AND SET
*                                      MODIFIED DATA TAG (MDT) BIT
         EXIT MODULE=GENEXIT,PARM=(10) INSERT DATA INTO FIELD
         WAIT
         ENDTXT
```

ITPFIOX File I/O User Exit

ITPFIOX is a TPNS message generation exit (EXIT MODULE=ITPFIOX) and network control exit (NCTLEXIT=ITPFIOX) providing sequential file I/O support to TPNS scripts.

QSAM is used to perform the actual file I/O operations. Storage for the DCB is allocated below the 16 MB line. The file handle, DCBE, and other control blocks are allocated above the 16 MB line. The DCBE is coded with RMODE31=BUFF which allows QSAM to allocate the block buffers above the 16 MB line. GET locate and PUT locate modes are used to access the record data.

Data sets must be preallocated and partitioned data sets are supported through member reference. The exit dynamically allocates the DD statement required for the data set unless the data set name is specified as DDNAME=*ddname* using an existing DD statement. Existing DCB attributes are used when data sets are opened for input. DCB attributes are set when data sets are opened for output.

Once a file handle is available after an OPEN, any simulated device in any active network can issue file I/O requests by passing the file handle value to the user exit.

ITPFIOX Syntax and Use

The syntax for ITPFIOX is as follows:

EXIT MODULE=ITPFIOX,
PARM=*(file_request rc_counter handle_sa# [data_sa#] [recfm] [blksize] [lrecl])*

Where:

- *file_request* is one of the following:

  OPENI Open the data set for input

  OPENO Open the data set for output

  OPENA Open the data set for output append

  READ Read record

  WRITE Write record

  CLOSE Close the data set

- *rc_counter*, the return code counter name, is one of the following:

  DC1-DC4095

  NC1-NC4095

  The counter contains the return code after each file I/O request completes. The following codes are set:

  0 OK

  1 Record truncated, READ or WRITE

  2 READ EOF

  8 Parameter error

  12 Request error

16	Handle error
20	Data error
24	DD allocation error
28	GETMAIN error
32	OPEN error
36	READ error
40	WRITE error
44	CLOSE error
48	Allocate data save area error
52	DD clear error
56	Find data save area error
60	Interface exit routine error
64	DCB values(*recfm, blksize*, or *lrecl*) error
68	Invalid member name
72	Data set not found
76	Data set in use
80	Invalid data set name
84	DD allocation, unexpected error
88	DSORG error, PO without member or PS with member
92	OPENA(append) for member of PDS
96	WRITE null record
100	DD name error

- *handle_sa#*, the file handle save area name, is one of the following:

  1-4095

  N1-N4095

  A four-byte address is saved in the handle save area when the file is opened. This value must be returned in the specified save area for all the other file I/O requests. The value is validated to avoid errors.

- *data_sa#*, the data save area name, is one of the following:

  1-4095

  N1-N4095

  For OPENI, OPENO, and OPENA, this save area contains the MVS data set name or DDNAME = *ddname* keyword.

  For WRITE, this save area contains the record data to be written.

  For READ, this save area contains the record read from the data set.

  For CLOSE, this save area is not required.

- *recfm* is either **VB** or **FB**

- *blksize* is **1-32760**
- *lrecl* is **1-32760**

Note:

1. When data sets are opened for output, the following DCB default values are set if *recfm, blksize,* and *lrecl* are not specified.

   RECFM=VB   BLKSIZE=23476   LRECL=23472

2. For VB, *lrecl* can be maximum of four bytes less than *blksize*
3. For FB, *blksize* must be a multiple of *lrecl*
4. Code NCTLEXIT=ITPFIOX and the open files in a network will be closed when the network is cancelled or reset.
5. OPENA (append) is not accepted for a partitioned data set.
6. When DDNAME=*ddname* is specified as the data set name, the file is OPENed against the DD name specified without allocating a DD statement. OPENO and OPENA are equal because the disposition of the data set is controlled by the DISP=value on the DD statement.
7. DISP=SHR is set on the DD statement for data sets OPENed for input (DDNAME=*ddname* not specified).
8. DISP=OLD is set on the DD statement for data sets OPENed for output (OPENO and DDNAME=*ddname* not specified).

The following are examples of the ITPFIOX syntax:

```
DATASAVE AREA=254,TEXT=(MYMVS.FILE)
SET DC55=999
EXIT MODULE=ITPFIOX,PARM=(OPENO DC55 252 254)
IF WHEN=IMMED,LOC=DC55,COND=NE,TEXT=0,THEN=B-ERROR

DATASAVE AREA=254,TEXT=(RECORD 1 DATA)
EXIT MODULE=ITPFIOX,PARM=(WRITE DC55 252 254)
IF WHEN=IMMED,LOC=DC55,COND=NE,TEXT=0,THEN=B-ERROR

EXIT MODULE=ITPFIOX,PARM=(CLOSE DC55 252)
IF WHEN=IMMED,LOC=DC55,COND=NE,TEXT=0,THEN=B-ERROR

DATASAVE AREA=254,TEXT=(MYMVS.FILE)
SET DC55=999
EXIT MODULE=ITPFIOX,PARM=(OPENI DC55 252 254)
IF WHEN=IMMED,LOC=DC55,COND=NE,TEXT=0,THEN=B-ERROR

EXIT MODULE=ITPFIOX,PARM=(READ DC55 252 254)
IF WHEN=IMMED,LOC=DC55,COND=NE,TEXT=0,THEN=B-ERROR
WTO (FIRST RECORD READ = $RECALL,254$)

EXIT MODULE=ITPFIOX,PARM=(READ DC55 252 254)
IF WHEN=IMMED,LOC=DC55,COND=NE,TEXT=2,THEN=B-ERROR

EXIT MODULE=ITPFIOX,PARM=(CLOSE DC55 252)
IF WHEN=IMMED,LOC=DC55,COND=NE,TEXT=0,THEN=B-ERROR
```

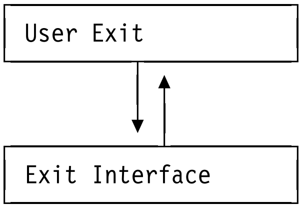
STL Language Example

The following is an example of ITPFIOX in STL:

```
constant handle_sa#   '253'
constant data_sa#     '254'
constant rc_counter#  'DC55'

constant rc_eof        2

allocate file_handle handle_sa#
allocate file_data   data_sa#
allocate file_rc     rc_counter#

file_data = 'MYMVS.FILE'
file_rc = 999
userexit('ITPFIOX', 'OPENO' rc_counter# handle_sa# data_sa#)
if file_rc <> 0 then call error

file_data = 'Record 1'

userexit('ITPFIOX', 'WRITE' rc_counter# handle_sa# data_sa#)
if file_rc <> 0 then call error

userexit('ITPFIOX', 'CLOSE' rc_counter# handle_sa#)
if file_rc <> 0 then call error

file_data = 'MYMVS.FILE'
file_rc = 999
userexit('ITPFIOX','OPENI' rc_counter# handle_sa# data_sa#)
if file_rc <> 0 then call error

userexit('ITPFIOX','READ' rc_counter# handle_sa# data_sa#)
if file_rc <> 0 then call error
say 'Record 1 =' file_data

userexit('ITPFIOX','READ' rc_counter# handle_sa# data_sa#)
if file_rc <> rc_eof then call error

userexit('ITPFIOX','CLOSE' rc_counter# handle_sa#)
if file_rc <> 0 then call error
```

Exit Interface Routine

User exit routines can call the exit interface routine to request WSim to perform certain functions or to return information to the user exit. The sequence of events is shown below.



Note: VREXIT cannot access the Exit Interface Routine.

The following types of functions are performed by the interface routine:

- Print EBCDIC data
- Print hexadecimal data
- Log EBCDIC data
- Log hexadecimal data
- Issue an operator command with or without completion notification
- Issue a WTO
- Manage network and device save areas
- Provide information.

When the interface routine is called, the registers must be set as follows:

Register	**Usage**
1	Address of the parameter list outlined below
13	Address of a standard 18 fullword save area
14	Return address
15	Address of the interface routine (passed to exit in parameter list)

The parameter list addressed by register 1 must contain the following addresses set up by the user exit routine:

Word 1 The address of the DEV control block.

Word 2 Address of a single byte indicating the type of request. The byte must be set to one of the following.

Requests for functions:

X'01' Print EBCDIC data.

X'02' Print hexadecimal data.

X'03' Write EBCDIC informational data to log data set.

X'04' Write hexadecimal informational data to log data set.

X'05' Issue an operator command.

X'06' Issue a write to operator (WTO).

X'07' Issue an operator command and schedule the UXOCEXIT routine to be executed upon completion.

Requests for information or network or device save area management:

X'50' The address of the network name (NCBNAME). This name is located in an eight-byte field, left-justified and blank-padded.

X'51' The address of the network sequence counter (NCBSEQ). Each counter is four bytes long.

X'52' The address of the network index counters (NCBSEQCT). Each counter is four bytes long.

X'53' The address of the network user area (NCBUSER) and the address of the length of network user area (NCBUSRLN). Lengths and counts are two-byte binary numbers.

X'54' The address of the network switches (NCBSWCH). This is a four-byte field containing 32 switches, numbered left to right.

X'55' The address of the number of index counters (NCBCNTRS) allocated for each level. This is a one-byte field.

Note: This can be either the value specified on the CNTRS= operand or the highest counter referenced in the network. The greater of the two values is used.

X'56' The address of a specific network save area, the address of the length of the data in the network save area, and the address of the length of the network save area. The lengths are two-byte binary numbers.

Note: The user exit must place the network save area number (1-255) in the first byte of the return area (see Word 3) before calling the exit interface routine

X'57' This request allocates a new network save area for the size requested and frees the existing network save area if currently allocated. The user exit must place the network save area number (1-255) in the first byte of the return area (see Word 3) and the desired network save area size (1-32767) in the second and third bytes of the return area. If the network save area allocation is successful, the X'56' request information is returned.

X'58' This request frees a currently allocated network save area. The user exit must place the network save area number (1-255) in the first byte of the return area (see Word 3).

X'59' This request increases the size of a currently allocated network save area while retaining any saved data in the increased network save area. The user exit must place the network save area number (1-255) in the first byte of the return area (see Word 3) and the desired network save area size (1-32767) in the second and third bytes of the return area. If the network save area increase is successful, the X'56' request information is returned.

X'5A' This request returns the address of a specific network save area, the address of the length of the data in the network save area, and the address of the length of the network save area. The lengths are two-byte binary numbers.

Note: The user exit must place the network save area number (1-4095) in the first two bytes of the return area before calling the exit interface routine.

X'5B' This request allocates a new network save area for the size requested and frees the existing network save area if currently allocated. The user exit must place the network save area number (1-4095) in the first two bytes of the return area and the desired network save area size (1-32767) in the third and fourth bytes of the return area. If the network save area allocation is successful, the X'5A' request information is returned.

X'5C' This request frees a currently allocated network save area. The user exit must place the network save area number (1-4095) in the first two bytes of the return area.

X'5D' This request increases the size of a currently allocated network save area while retaining any saved data in the increased network save area. The user exit must place the network save area number (1-4095) in the first two bytes of the return area and the desired network save area size (1-32767) in the third and fourth bytes of the return area. If the network save area increase is successful, the X'5A' request information is returned.

X'5E' This request returns the address of the number of network index counters. This is a two-byte field.

X'5F' This request returns the address of the number of network switches. This is a two-byte field.

X'70' The address of the line name (LINNAME). This name is located in an eight-byte field, left-justified and blank-padded.

X'71' The address of the line sequence counter (terminal simulation: LINSEQ; cross-domain: TRMSEQ). Each counter is four bytes long.

X'72' The address of the line index counters (terminal simulation is LINSEQCT and cross-domain is TRMSEQCT). Each counter is four bytes long.

X'73' The address of the line ID. This ID is six EBCDIC characters located in a six-byte field.

X'74' This request returns the address of the number of line index counters. This is a two-byte field.

X'90' The address of the terminal name (TRMNAME). This name is located in an eight-byte field, left-justified and blank-padded.

X'91' The address of the terminal sequence counter (TRMSEQ). Each counter is four bytes long.

X'92' The address of the terminal index counters (TRMSEQCT). Each counter is four bytes long.

X'93' The address of the terminal switches (TRMSWCH). This is a four-byte field containing 32 switches, numbered left to right.

X'94' This request returns the address of the number of terminal index counters. This is a two-byte field.

X'95' This request returns the address of the number of terminal switches. This s a two-byte field.

X'B0' The address of the device name (DEVNAME). This name is located in an eight-byte field, left-justified and blank-padded.

X'B1' The address of the device sequence counter (DEVSEQ). Each counter is four bytes long.

X'B2' The address of the device index counters (DEVSEQCT). Each counter is four bytes long.

X'B3' The address of the device user area (DEVUSRAD) and the address of the length of device user area (DEVUSRLN). Lengths and counts are two-byte binary numbers.

X'B4' The address of the device switches (DEVSWCH). This is a four-byte field containing 32 switches, numbered left to right.

X'B5' The address of a specific device save area, the address of the length of the data in the device save area, and the address of the length of the device save area. The lengths are two-byte binary numbers.

Note: The user exit must place the save area number (1-255) in the first byte of the return area (see Word 3) before calling the exit interface routine.

X'B6' The address of a one-byte device type (DEVTYPE).

X'B7' For display devices, the address of the display buffer and the address of the length of the display buffer. For 3270 devices, this is PTNPSBUF and PTNPSSIZ. For other display devices, this is DEVOTBUF and DEVOBUFL. Lengths and counts are two-byte binary numbers.

For non-display devices, the address of the message generation buffer and the address of the length of message generation buffer (DEVOTBUF/DEVOBUFL). Lengths and counts are two-byte binary numbers.

X'B8' The address of the 3270 attribute table (PTNATRTB) and the address of the attribute count for 3270 devices (PTNATRCT). Lengths and counts are two-byte binary numbers.

X'B9' The address of the format table for LU7 devices (DEVFTBAD) and the address of the length of the format table (DEVFTBLN) for LU7 devices. Lengths and counts are two-byte binary numbers.

X'BA' For display devices, the address of the cursor location. This is a two-byte binary number representing the relative position of the cursor on the simulated screen, starting with 1. For example, the first screen position (row 1, column 1) would be 01. The third screen position (row 1, column 3) would be 03. For 3270 devices, this is PTNCCP. For others, this is DEVCURSR.

For non-display devices, the address of the message generation data length (a two-byte binary number representing the length of data currently in the message generation buffer) (DEVCURSR).

X'BB' The address of the active 3270 PTN control block (DEVACPTN).

X'BC' The address of the current AID byte (DEVAID).

X'BD' The address of the log byte (DEVUSER).

X'BE' The address of the LU session number or TP instance number (a five-byte field containing the EBCDIC representation of the current LU session number or TP instance number) (DEVSESNO).

X'BF' This request allocates a new device save area for the size requested and frees the existing device save area if currently allocated. The user exit must place the device save area number (1-255) in the first byte of the return area (see Word 3) and the desired device save area size (1-32767) in the second and third bytes of the return area. If the device save area allocation is successful, the X'B5' request information is returned.

X'C0' This request frees a currently allocated device save area. The user exit must place the device save area number (1-255) in the first byte of the return area (see Word 3).

X'C1' This request increases the size of a currently allocated device save area while retaining any saved data in the increased device save area. The user exit must place the device save area number (1-255) in the first byte of the return area (see Word 3) and the desired device save area size (1-32767) in the second and third bytes of the return area. If the device save area increase is successful, the X'B5' request information is returned.

Note: If a terminal control block (TRM) is passed to the user interface routine, this control block will be used to provide information for DEV requests.

X'C2' This request returns the address of a specific device save area, the address of the length of the data in the device save area, and the address of the length of the device save area. The lengths are two-byte binary numbers.

Note: The user exit must place the device save area number (1-4095) in the first two bytes of the return area before calling the exit interface routine.

X'C3' This request allocates a new device save area for the size requested and frees the existing device save area if currently allocated. The user exit must place the device save area number (1-4095) in the first two bytes of the return area and the desired device save area size (1-32767) in the third and fourth bytes of the return area. If the device save area allocation is successful, the X'C2' request information is returned.

X'C4' This request frees a currently allocated device save area. The user exit must place the device save area number (1-4095) in the first two bytes of the return area.

X'C5' This request increases the size of a currently allocated device save area while retaining any saved data in the increased device save area. The user exit must place the device save area number (1-4095) in the first two bytes of the return area and the desired device save area size (1-32767) in the third and fourth bytes of the return area. If the device save area increase is successful, the X'C2' request information is returned.

X'C6' This request returns the address of the number of device index counters. This is a two-byte field.

X'C7' This request returns the address of the number of device switches. This is a two-byte field.

Note: Return code 44 is set when the save area number is greater than 4095.

Word 3 Word 3 contains the following information, depending on the request:

- For function request X'01' through X'06', Word 3 contains the address of a six-byte parameter describing the data to be processed. The first four bytes contain the address of the data. The next two bytes contain the length of the data. Data to be printed can be up to 32767 bytes long. Data to be logged can be up to 32767 bytes long; however, a maximum of 5080 bytes is actually logged. Data for a WTO can be up to 111 bytes long. Data for an operator command can be up to 32767 bytes long, but only the first 120 bytes is used.
- For function request X'07', Word 3 contains the address of a 12-byte parameter describing the data to be processed. The first four bytes contain the address of the data. The next two bytes contain the length of the data. The next two bytes contain flags that are formatted as follows:

```
10...... ........  monitor the command regardless of the NTWRK
                   option
.1...... ........  do not monitor the command regardless of the
                   NTWRK option
00...... ........  use the NTWRK option to determine whether to
                   monitor the command
```

The final four bytes contain a user word to be returned in the first word of the parameter list passed to the UXOCEXIT when this command ends.

- For an information or network or device save area management request, Word 3 must contain the address of a one to three word return area, depending on the request. If only one address is returned, only the first word will be used. Any requests that return two or three addresses will insert these addresses into the successive words.

Return Codes

When the interface routine returns control to the user exit, register 15 is set to one of the following codes:

Code **Description**

0 Normal completion.

4 Invalid request code.

8 Invalid control block passed (Word 1) for request type.

12 Incorrect data length specified.

16 Insufficient storage available for request.

20 Unable to supply requested information for one of the following reasons:

- The referenced static save area is not allocated, the referenced dynamic save area is empty (not allocated), or the requested save area cannot be found.
- The referenced NTWRK or DEV user area does not exist.
- The referenced LINE counters do not exist.
- The referenced LINE name does not exist.
- The referenced LINE value does not exist.
- The referenced TERM does not exist.
- The referenced AID byte does not exist for a non-display terminal.
- The referenced session number does not exist.
- The referenced attribute table does not exist.
- You referenced a LU7 Format Table and this is not a LU7.
- The referenced LU2 PTN does not exist.

24 Save area number zero was specified for an allocate, free, or increase network or device save area request.

28 The specified network or device save area could not be found for a free or increase request.

32 The device save area was allocated statically using the SAVEAREA operand in the WSim network definition and cannot be allocated, freed, or increased.

36 The save area size value specified for a network or device save area allocate or increase request is invalid. The value must be within the range of 1-32767.

40 No save area storage was available for a network or device save area allocate or increase request.

Sample JCL Definitions

JCL for a sample assembly and link-edit for a user exit program on MVS is shown below.

```
//EXITJOB JOB
//ASM      EXEC PGM=IFOX00,PARM='LOAD,NODECK'
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD UNIT=SYSDA,SPACE=(TRK,(20,20))
//SYSUT2   DD UNIT=SYSDA,SPACE=(TRK,(20,20))
//SYSUT3   DD UNIT=SYSDA,SPACE=(TRK,(20,20))
//SYSLIN   DD UNIT=SYSDA,DSN=&OBJ,SPACE=(TRK,(5,5)),
//            DISP=(,PASS)
//SYSLIB   DD DSN=SYS1.MACLIB,DISP=SHR
//SYSIN    DD *
  ⋮
assembler source statements
  ⋮
//LKED     EXEC PGM=LINKEDIT
//SYSPRINT DD SYSOUT=A
//SYSLIN   DD DSN=&OBJ,DISP=(OLD,DELETE)
//SYSUT1   DD UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSLMOD  DD DSN=WSIM.SITPUEX(userexit),DISP=OLD
```

Note: Under MVS, concatenate WSIM.SITPUEX with WSIM.SITPLOAD on the JOBLIB or STEPLIB DD statement in the JCL of your MVS procedure, batch job, or TSO logon procedure when executing WSim or the preprocessor, and authorize this data set to the operating system if WSIM.SITPLOAD was authorized. However, if the SYSLMOD DD statement was changed to DSN=WSIM.SITPLOAD, neither of these two steps would be necessary. The user exit would then reside in the WSim load library.

Compatible Exits

Since many of the exits have compatible parameter lists (INEXIT, OUTEXIT, NCTLEXIT, UCMDEXIT, NETEXIT, INFOEXIT, message generation EXIT statement, and UXOCEXIT), you can write a general exit routine to handle a variety of situations. These exits can be differentiated by the exit flags contained in word 4 of the parameter lists.

Exits named on an EXIT operand cannot be named on other types of exit operands. That is, an exit named on an EXIT operand cannot be named anywhere else in the same network and can only be named on an EXIT operand in another network. The reason for this restriction is that this exit has unique parameter lists.

Operational Suggestion

User exit programs to modify WSim internal control blocks, such as the device control block (DEV) or partition control block (PTN), are not required in most cases. For display devices, use the user exit programs that modify the internal control blocks to maintain the display resources (such as display buffer, attribute table, and extended attribute buffer) in the same manner as WSim.

The best way to allow WSim to maintain the resources of the simulated terminal and give the user exit program control of the message generation process is to maintain a defined interface between WSim and the user exit program in the device user area or save area. The user exit program places control information and message generation data into predefined locations of a user area or save area. You can use IF statements within the WSim message generation deck to test the predefined control fields in the user area and also to generate data from the user area or save area using the $RECALL$ data field option.

Performance Considerations

When you invoke WSim user exits, you must pay attention to the potential performance degradation that can occur due to the user exit code itself. At least two performance problems can arise:

- Message traffic rates can decline.
- Real telecommunication lines can timeout.

Either of these situations can occur if user exits are not efficiently designed. The exit may, unknown to you, consume much more real time than expected, causing the main WSim task to be placed into a wait state for too long.

Although not always problems, the following user exit situations can cause performance degradations and should be avoided if you suspect a WSim performance degradation is due to the user exit:

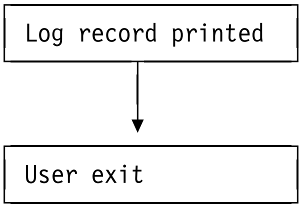
- Use of high-level languages, such as PL/I, and COBOL
- Excessive WTOs issued within the user exit
- Disk, tape, or other input/output done directly
- Other operating system calls or supervisor calls (SVCs).

Chapter 2. Coding Loglist Utility User Exit Routines

This chapter describes how to code exit routines for the Loglist Utility. It discusses parameter lists and register linkage, tells how to interpret the various types of records in the log data set, and provides a sample Loglist Utility user exit routine.

Loglist User Exit Routines

The Loglist Utility (see *WSim Utilities Guide* for more information) provides an interface through which a user routine can gain control and perform any desired function using the log records. You can specify the exit routine on the EXIT Loglist Utility control command. The Loglist Utility calls the exit routine for each log record that satisfies the other input command specifications. The exit routine does not replace any normal Loglist Utility processing. It is called after the Loglist Utility formats and prints a selected record. The sequence of events is shown below.



Registers must be saved upon entering the exit routine and restored before returning to the Loglist Utility. The registers contain the following when the exit routine is entered:

Register	**Contents**
1	Address of a 3-word parameter list
13	Address of an 18 fullword save area
14	Return address
15	Address of the exit routine being called.

The parameter list passed with register 1 consists of the following three fullword pointers:

Word 1 Address of the record read from the log data set. This word will be zero (X'00000000') on the last call to the exit for each RUN. This should be used by the exit to perform final processing and clean-up activities, such as closing files it might have opened.

Word 2 Address of a print control block (PRT). Refer to “Print Control Block (PRT)” on page 70 for the structure of this control block.

Word 3 Address of a fullword that contains the address of the WSim print routine.

To print data on the Loglist Utility (ITPLL) SYSPRINT data set from the exit routine, move the data into the print line (PRTPRTLN) of the print control block (PRT) and call the print routine (for example, BALR 14,15). The print line consists of 133 characters. The first character is used by the print routine as a printer carriage control character. The print routine fills the print line with blanks after printing it and then returns to the exit routine.

Note: Do not modify any fields in the print control block (PRT) other than the print line (PRTPRTLN).

When the WSim print routine is called, register 1 must point to a 1-word parameter list that contains the address of the print control block (PRT).

Analyzing the WSim Log Data Set

The WSim log data set provides a history in time sequence of the activity that occurred during a simulation run. This section describes the types of records that are on the log data set and discusses how to interpret them as part of a simulation run analysis.

Log Data Set Records

Each record on the WSim log data set contains an 88-byte header followed by the data transmitted or received, an informational message, or trace data. The maximum length of the data portion of the record is specified by the MLEN operand in the network definition. For more information on the format of the log data set record, see “Log Record Header Format (LOG)” on page 64.

The log record header contains a field that identifies the type of record being logged. The record type can be one of the following:

Console record
A console record contains either an operator command or an operator command response in the data portion of the record. Operator commands, whether entered at the WSim operator console or issued with OPCMND message generation statements, are always logged.

CPI-C trace record
A CPI-C trace record contains general messages that trace the execution of CPI-C transaction programs (TPs).

Informational record
An informational record is written to the log data set when errors occur during a simulation run, when an action or state change is accomplished by WSim, or when a user exit invokes the WSim interface routine for logging data.

Log record
A log record is written to the log data set whenever a LOG statement or a LOG operand of an IF statement is encountered during message generation. The data is written in an interpreted dump format so that it appears in hexadecimal and EBCDIC.

Log display record
A log display record is written to the log data set each time a simulated 3270 or 5250 display/printer image buffer is written to the log data set as a result of the LOGDSPLY operand or the LOG DISPLAY statement. See “Log Display Record Header Format (LDS)” on page 62 for more information about these records.

Marker record
A marker record is written to the log data set each minute of the simulation run. This record has a header only. It contains no data.

Message data record
A message data record contains data sent or received by a WSim simulated resource.

Message trace record
A message trace record contains general messages about the message generation path through decks as well as the IF messages about logic tests.

STL trace record
An STL trace record contains general messages about the flow of execution through STL programs.

Verify data record
A verify data record contains information relevant to IF statements processed with the VERIFY action specified.

Using the Loglist Utility to Read the Log Data Set

You can use the Loglist Utility to read the WSim log data set, without using the other features of this utility. If you want to write your own log analysis program, you can let the Loglist Utility read the log, assemble segmented records, and (optionally) select specific record types to process. These records can then be passed to the user-written analysis program using the Loglist Utility EXIT facility.

If you use the Loglist Utility in this way and you do not want the normal output generated by the Loglist Utility, specify DUMMY for the SYSPRINT DD statement. If output from the Loglist Utility exit routine is to be routed to a file other than the SYSPRINT data set, include a DD identifying that file in the Loglist Utility invocation JCL. The exit routine is responsible for opening and closing such files.

The example below shows a simple Loglist user exit routine that uses the Loglist Utility to read the log data set. This routine writes the data portion of each log record to a file defined by DD EXITFILE. The file is opened on the first call to this routine and closed on the last call.

Note: This exit works in 24- or 31-bit addressing mode. You must link-edit this with RMODE(24) so the actual code and data areas reside in 24-bit addressable storage.

```
ITPLLXIT CSECT                       ITPLL EXIT TO READ LOG DATA SET
         STM   14,12,12(13)          SAVE CALLER'S REGISTERS
         BALR  12,0                  SET UP BASE REGISTER
         USING *,12                  TELL ASSEMBLER ABOUT IT
         USING LOG,4                 LOG DSECT FROM SAMPLES DATA SET
         ST    13,SAVEA+4            SAVE CALLER'S SAVEAREA ADDR
         LA    14,SAVEA              ADDR OF OUR SAVEAREA
         ST    14,8(,13)             SAVE IN CALLER'S SAVEAREA
         LR    13,14                 ADDR OF OUR SAVE AREA

*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
*  WAS THIS CALL TRIGGERED BY END OF FILE CONDITION?                  *
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
         L     4,0(,1)               ADDR OF "LOG" RECORD OR EOF (0)
         LTR   4,4                   CHECK FOR ZERO PARM
         BZ    CLOSEIT               NO RECORD, CLOSE OUTPUT FILE
```

```
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
*  HERE IF A LOG RECORD WAS PASSED TO THE EXIT.  IF THIS IS THE FIRST *
*  TIME THE EXIT HAS BEEN CALLED, OPEN THE OUTPUT FILE.                *
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
         TM    SWITCHES,FIRSTIME    FIRST TIME CALLED?
         BZ    PUTREC               NOT FIRST TIME - DON'T OPEN
         OPEN (PRINTER,(OUTPUT))    MACRO - OPEN OUTPUT FILE
         NI    SWITCHES,FF-FIRSTIME RESET FOR SUBSEQUENT CALLS

*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
*  WRITE THE LOG RECORD DATA TO THE EXIT OUTPUT FILE                   *
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
PUTREC   LH    5,LOGLENG            GET LENGTH OF THE LOG DATA
         LA    2,132                MAX LENGTH IS 132 BYTES
         CR    2,5                  USE WHICHEVER IS LOWER
         BNH   BLANKS               LOGLENG >= 132
         LR    2,5                  LOGLENG < 132 - USE IT
BLANKS   MVI   DATA,C' '            BLANK OUT DATA AREA
         MVC   DATA+1(132),DATA       "    "   "     "
         LTR   5,5                  IS THERE REALLY LOG DATA?
         BZ    OUTPUT               NO DATA - OUTPUT BLANK LINE

*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
*  HERE IF THERE IS DATA TO PRINT                                      *
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
         LR    5,2                  LENGTH OF DATA
         BCTR  5,0                  DECREMENT LENGTH FOR MVC
         EX    5,MOVEDAT            EXECUTE MOVE

*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
*  ISSUE THE QSAM PUT IN 24 BIT ADDRESSING MODE                        *
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
OUTPUT   LA    6,127                CHECK
         SLL   6,24                       FOR
         LA    6,0(,6)                        31 BIT
         LTR   6,6                                   ADDRESSING
         BNZ   AMODE31              BRANCH IF 31 BIT ADDRESSING
         PUT PRINTER,DATA           MACRO - WRITE TO OUT FILE
         B     RETURN               PROCESSING COMPLETE
AMODE31  LA    7,PUT24BIT           ADDRESS OF PUT ROUTINE
         BASSM 6,7                  CALL AND SET 24 BIT ADDRESSING
         B     RETURN               PROCESSING COMPLETE
PUT24BIT PUT PRINTER,DATA           MACRO - WRITE TO OUT FILE
         BSM   0,6                  RETURN AND SET 31 BIT ADDRESSING

*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
*  HERE IF EXIT CALLED TO CLOSE OUTPUT FILE                            *
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
CLOSEIT  CLOSE PRINTER              MACRO - CLOSE OUTPUT FILE
         OI    SWITCHES,FIRSTIME    PREPARE FOR NEXT ITPLL RUN
RETURN   L     13,4(,13)            CALLER'S SAVEAREA ADDR
         LM    14,12,12(13)         RESTORE REGS
         BR    14                   RETURN
```

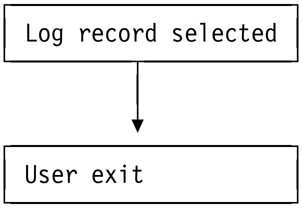
```
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
*  DEFINITIONS                                                        *
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
SAVEA    DS    18F                    SAVE AREA
MOVEDAT  MVC   DATA+1(0),LOGDATA      EXECUTED TO MOVE LOG DATA
SWITCHES DC    AL1(FIRSTIME)          LOCAL SWITCHES
FIRSTIME EQU   X'80'                  FIRST TIME THROUGH SWITCH
FF       EQU   X'FF'                  X'FF' FOR ANDING BITS OFF
DATA     DC    CL133' '               LOCAL COPY OF LOG DATA
PRINTER  DCB DSORG=PS,MACRF=(PM),DDNAME=EXITFILE,RECFM=FBA,LRECL=133
LOG      DSECT
         ORG   LOG+60
LOGLENG  DS    0H                     LENGTH OF LOG DATA
         ORG   LOG+88
LOGDATA  DS    0C                     LOG DATA
         END
```

Chapter 3. Coding Response Time Utility User Exit Routines

This chapter describes how to code exit routines for the Response Time Utility. It discusses parameter lists and register linkage and provides a sample Response Time Utility user exit routine.

Response Time Utility User Exit Routines

The Response Time Utility allows user routines to gain control with the EXIT command and perform any desired functions using the log records. If you do not code a user exit and call it with the EXIT command, the WSim-supplied exit routine processes the log records and calculates the values for the standard output reports according to the previously defined rules. If you do code a user exit with the EXIT command, it receives control for each log record that satisfies the input command specifications. The sequence of events is shown below.



Registers must be saved on entry to the exit and then restored prior to returning to the Response Time Utility. The registers contain the following when the exit routine is entered:

Register	**Contents**
1	Address of a 2-word parameter list
13	Address of an 18 fullword save area
14	Return address
15	Address of the exit routine being called.

The parameter list passed with register 1 consists of the following two fullword pointers:

Word 1 Address of the user parameter data specified by the PARM operand on the EXIT command. The first two bytes of the user parameter data contain the length of the data coded in the PARM operand. The actual data coded directly follows the length field. If the PARM operand was not coded, the length field contains binary zero.

Word 2 Address of the Response Time control block (RSP). This control block contains the address of the log record and flags that the user exit can set. Table 2 on page 48 describes the fields in the RSP corresponding to the parameters that are passed to the user exit routine by the Response Time Utility.

When a user exit is invoked for a log record, the RSPIGNOR bit in the RSPFLAG3 field is set to B'1', indicating that the record is to be ignored by the WSim-supplied exit. The user exit can then set this bit to B'0', which allows the WSim exit to subsequently process the record. This way, a user exit can apply additional selection criteria to the log records selected by the Response Time Utility input commands, possibly by using the PARM data.

When end of file is encountered on the input log data set, the user exit is called with register 1 pointing to a fullword containing binary zeros. This final call to the user exit allows for final calculations and report printing. If the exit routine returns to the Response Time Utility with a return code of zero in register 15, the Response Time Utility prints the normal WSim reports. If the return code is not zero, the reports are not printed.

Table 2. RSP Fields that Correspond to WSim V1R1 Parameter List

RSP field	Offset	Length	Description
RSPBUFAD	0 (0)	4	Address of the log record

RSPFLAG3	233 (E9)	1	Flags defined as follows:
			`x... .... RSPACTUL`
			`          0=PROCESS SYSTEM`
			`          1=PROCESS ACTUAL`
			`.xxx xxx. Reserved`
			`.... ...x RSPIGNOR`
			`          0=record processed`
			`          1=record ignored`

The code below shows an example of a user exit for the Response Time Utility.

```
    TITLE   'OMIT SELECTED TRANSACTIONS FROM ITPRESP REPORTS'
OMITZERO CSECT
*
*   This is a user exit routine that looks for transactions
*   in WSim logfiles (during the execution of ITPRESP) that have
*   the characters "NO" in the tenth and eleventh
*   characters of the XMIT record.
*   All transactions meeting this criteria are ignored.
*
         STM     14,12,12(13)
         BALR    12,0
         USING   *,12
         ST      13,SAVEAREA+4
         LA      14,SAVEAREA
         ST      14,8(13)
         B       @PGM
         DC      C'OMIT ZERO RESPONSE TIMES MODULE'
@PGM     L       2,0(,1)           CHECK FOR LAST RECORD
         LTR     2,2               SET CONDITION CODE
         L       3,4(,1)           LOAD ADDRESS OF RSP CONTROL BLOCK
         BZ      PROCESS           BRANCH ON LAST RECORD
         L       4,0(,3)           LOAD ADDRESS OF LOG HEADER RECORD
         CLC     97(2,4),NOPE      ARE THE TWO CHARS 'NO'
         BE      BADONE            BRANCH IF A ZERO RESPONSE
PROCESS  NI      233(3),X'FE'      SET BIT OFF TO PROCESS TRANSACTION
         B       END
BADONE   OI      233(3),X'01'      SET BIT ON TO IGNORE TRANSACTION
END      L       13,SAVEAREA+4     RESTORE REGS TO THE WAY THEY WERE
         LM      14,12,12(13)
         LA      15,0              REG15 --> RETURN CODE
         BR      14                EXIT THE PGM
NOPE     DC      CL2'NO'
SAVEAREA DS      18F
         END
```

Appendixes

Appendix A. Network-Level Exit (EXIT Operand)

The following section provides a description of the addresses found in the parameter list for the network-level user exit specified using the EXIT operand. A sample exit program using the EXIT operand is also provided for further understanding.

Note: This is an obsolete alternative to the INEXIT, OUTEXIT, and NETEXIT operands. When EXIT is coded, the module gets control when data is received or transmitted at the terminal level. The EXIT operand is mutually exclusive of the INEXIT, OUTEXIT, and NETEXIT operands. It is recommended that you use INEXIT, OUTEXIT, or NETEXIT whenever possible.

Parameter List

Register 1 contains the address of the parameter list for the network-level user exit specified using the EXIT operand. The parameter list consists of the following addresses:

Word 1 The address of the data sent or received. The data includes information sent or received, except for line control characters, if applicable. For SNA devices, the data begins with the TH field, followed by the RH and RU fields.

Word 2 The address of a 16-bit field containing the following indicators:

- **Bit 0** Set ON (B'1') for input to WSim.
  Set OFF (B'0') for output from WSim.
- **Bit 1** Set ON only by the user exit. If ON, processing on the current message stops. For output, the message will not be sent or logged. For input, the message will not be processed further. This bit is used for SNA terminal types only.
- **Bit 2** Set ON only by the user exit. If ON, the current message generation delay is canceled.
- **Bits 3-5** All of these bits will be in the OFF (B'0') state.
- **Bits 6-15** Reserved.

Word 3 The address of the halfword containing the length of data sent or received. The exit program can be used to change this field, but it cannot be larger than the length of the terminal buffer (pointed to by Word 4) and must not be set to zero.

Word 4 The address of the halfword containing the length of the terminal buffer. Do not change this field using the exit program.

Word 5 The address of a byte indicating the terminal or device type. See Appendix B, “Simulated Resource Type Codes” on page 53 for definition of these type values.

Word 6 The address of an eight-character field containing the name from the DEV or LU statement. The name is left-justified in the field and padded with blanks.

Word 7 The address of the user data field defined by the USERAREA parameter. The address defaults to zero if the area is not defined.

Word 8 The address of a halfword containing the length of the user data area. The address defaults to zero if the area is not defined.

Word 9 The address of the device control block (DEV) for the terminal, device, or logical unit associated with the current message. Refer to "Device Control Block (DEV)" on page 57 for the format of the DEV. See "Changing Device Parameters" on page 2 for more information.

Word 10 The address of a fullword containing the address of the WSim interface routine for the user exit. See "Exit Interface Routine" on page 30 for more information.

Sample Exit Program Using EXIT

The following sample program puts the terminal type into the first byte of all outgoing messages.

```
EXIT      CSECT                        NETWORK EXIT USING EXIT OPERAND
          SAVE  (14,12)                SAVE ALL REGISTERS
          BALR  12,0                   SET UP THE BASE REGISTER
          USING *,12
          L     REG2,4(REG1)           GET ADDR OF I/O BYTE
          TM    0(REG2),X'80'          IF ON,INPUT
          BO    BYPASS                 YES,BYPASS NEXT CODE
          L     REG2,0(REG1)           GET ADDR OF DATA
          L     REG11,16(REG1)         GET ADDR OF TERMINAL TYPE
          MVC   0(1,REG2),0(REG11) MOVE TERM CODE INTO MSG
BYPASS    RETURN (14,12)               RETURN TO WSim
REG2      EQU   2
REG11     EQU   11
REG15     EQU   15
REG1      EQU   1
          END
```

Appendix B. Simulated Resource Type Codes

This appendix lists the WSim code values for each simulated resource type that you can use to identify resource types when reading a log data set listing or processing in a user exit routine.

The following chart lists the code values for terminal resource types:

Terminal	**Type**	**Terminal**	**Type**
TCP/IP	30	VTAMAPPL	69

The following chart lists the code values for device resource types:

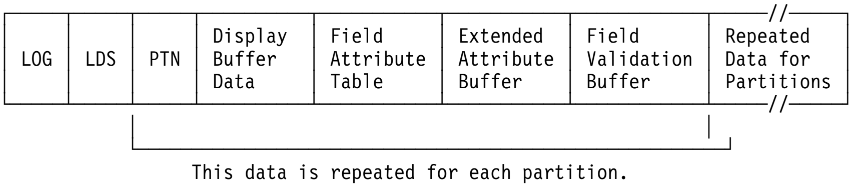
Device	**Type**	**Device**	**Type**
FTP (command conn)	91	LU0	E0
FTPD (data conn)	92	LU1	E1
STCP	93	LU2	E2
TN3270	94	LU3	E3
TN3270E	95	LU4	E4
TN3270P	96	LU6	E6
SUDP	97	LU7	E7
TNNVT	98	LU6.2	E9
TN5250	99	APPC TP	EA

Appendix C. Log Display Record Formats

When WSim simulates 3270 and 5250 terminals, log display records are written to the log data set whenever the LOG DISPLAY message generation deck statement is executed. These records can also be automatically logged during message generation depending on the value of the LOGDSPLY operand for these resources.

This appendix lists the format of the data in each log display record.

3270 Log Display Records



LOG	LDS	PTN	Display Buffer Data	Field Attribute Table	Extended Attribute Buffer	Field Validation Buffer	Repeated Data for Partitions

This data is repeated for each partition.

Figure 1. Log Display Record Format

One or more log display records can be written to the log data set to form a logical group of records that represent the simulated 3270 display image at the time of logging. Each log display record begins with a message log record header control block (LOG). For more information on the message log header, see "Log Data Set Records" on page 42.

Bits 17 (LOGFSTRC) and 18 (LOGLSTRC) in the flag bytes field of the LOG indicate if the record is the first or last record in a logical group of log display records. Both bits will be turned on when all the data is included in a single record. Bit 1 (LOGLDATA) in the flag bytes field of the LOG indicates data lost in the last block. User programs should monitor bit 1 along with bits 17 and 18 when processing log display records.

The first or only log display record of a logical group contains a log display header control block (LDS) that contains information about the device state and the data that follows. The Total Length field (LDSTOTLN) in the LDS contains the amount of storage required to hold the LDS and all of the data that follows the LDS in this record (and possibly other records) in this logical group.

The 3270 partition control block (PTN) follows the LDS. The PTN contains additional device state information, the size of the display buffer data, extended attribute buffer, and field validation buffer (PTNPSSIZ), and the size of the field attribute table (PTNATRSZ). The PTN is followed by the display buffer data and the field attribute table. When the LDSFECS bit is turned on in the LDS, an extended attribute buffer follows the field attribute table. When the LDSFFLDV bit is turned on in the LDS, a field validation buffer follows the extended attribute buffer.

When simulating a 3270 display terminal with partitions, the LDSPTNST bit in the LDS indicates that the device is in partition state. The active partition's PTN, display buffer data, field attribute table, extended attribute buffer, and field validation buffer immediately follow the LDS. The PTN, display buffer data, field attribute table, extended attribute buffer, and field validation buffer for all the inactive partitions follow the active partition's resources.

When a CLEAR key operation is performed, the log display record contains only the LOG and LDS.

See “Log Record Header Format (LOG)” on page 64 for the LOG control block, “Log Display Record Header Format (LDS)” on page 62 for the LDS control block and “Display Partition Control Block (PTN)” on page 71 for the PTN control block.

5250 Log Display Records

LOG	LDS	PTN	Display Buffer Data	Field Format Table

Figure 2. 5250 Log Display Record Format

A single log display record is written to the log data set to represent the simulated 5250 display image at the time of logging. Each log display record begins with a message log record header control block (LOG). For more information on the log record heading see “Log Data Set Records” on page 42. Bits 17 (LOGFSTRC) and 18 (LOGLSTRC) in the flag bytes field of the LOG will be turned on to indicate that all the data is included in this record.

The LOG is followed by the log display header control block (LDS) that contains information about the device state and the data that follows. The total length field (LDSTOTLN) in the LDS contains the amount of storage required to hold the LDS, and all of the data that follows the LDS, in this log display record.

The LDS is followed by the display buffer data and the field format table. The 5250 display buffer size field (LDS7BSZ) in the LDS contains the size of the display buffer data. The field format table is always 800 bytes long.

When a CLEAR key operation is performed, the log display record will contain only the LOG and LDS.

See “Log Record Header Format (LOG)” on page 64 for the LOG control block and “Log Display Record Header Format (LDS)” on page 62 for the LDS control block.

Appendix D. User Exit Control Blocks

This appendix lists the format of certain WSim control blocks which are provided in DSECT format in the WSIM.SITPUEX partitioned data set (MVS) on the WSim installation tape. Use only those fields documented in this appendix as a programming interface.

Device Control Block (DEV)

Size in bytes:	888 (X'378')
Pointed to by:	Chain based on TRMDEVAD Chain based on PULUAD
Function:	Contains information necessary to simulate a device, logical unit, or transaction program. A DEV is built for each DEV, LU, or TP statement in a network definition. If a TP has multiple instances, a DEV is built for each instance.

DEVICE CONTROL BLOCK

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	888	DEV	DEVICE CONTROL BLOCK
TRM-DEV FIELDS					
20	(14)	ADDRESS	4	DEVINBUF	ADDR OF INPUT BUFFER
24	(18)	ADDRESS	4	DEVOTBUF	ADDR OF OUTPUT BUFFER
28	(1C)	SIGNED	2	DEVOBUFL	LENGTH OF OUTPUT BUFFER
32	(20)	BITSTRING	1	DEVFLAG1	FLAG FIELDS
		.... .1..		DEVWAIT	WAIT BIT FROM LOGICAL COMPARE
		1		DEVEWAIT	WAITING ON EVENT
34	(22)	BITSTRING	1	DEVFLAG4	TRM/DEV FLAGS
		..1.		DEVQUIEC	DEVICE IN QUIESCE STATE
MESSAGE GENERATION FIELDS					
36	(24)	SIGNED	2	DEVCURSR	CURRENT CURSOR POSITION
52	(34)	CHARACTER	8	DEVNAME	CONTROL BLOCK NAME
64	(40)	ADDRESS	4	DEVCTRAD	TRM/DEV COUNTERS ADDRESS
68	(44)	BITSTRING	4	DEVSWCH	USER SWITCHES
85	(55)	ADDRESS	1	DEVUSER	USER DATA BYTE FOR LOGGING
STATISTICS					
200	(C8)	ADDRESS	4	DEVSAVAD	SAVEAREA BLOCK ADDRESS
212	(D4)	ADDRESS	4	DEVPTENT	CURRENT ENTRY IN PATH SEQUENCE
TIMER QUEUE ELEMENT					
228	(E4)	ADDRESS	4	DEVNCBAD	NETWORK CONTROL BLOCK ADDRESS
252	(FC)	ADDRESS	4	DEVLINAD	LINE CONTROL BLOCK ADDRESS
256	(100)	ADDRESS	4	DEVNXDEV	ADDRESS OF FIRST TRM/DEV
260	(104)	ADDRESS	4	DEVTRMAD	ADDRESS OF TERMINAL
INITIATOR FIELDS					
377	(179)	ADDRESS	1	DEVTYPE	TERMINAL TYPE
		1...		DEVDEV	TYPE IS FOR A DEVICE
		.1..		DEVSNA	SDLC TERMINAL
378	(17A)	SIGNED	2	DEVIBUFL	BUFFER SIZE
414	(19E)	SIGNED	2	DEVUSRLN	LENGTH OF USER DATA
428	(1AC)	ADDRESS	4	DEVUSRAD	POINTER TO USER AREA
432	(1B0)	ADDRESS	4	DEVNXTRM	ADDR OF NEXT TRM

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
DISPLAY ORIENTED FIELDS FOR 3270 AND LU7 TYPE DEVICES					
488	(1E8)	CHARACTER	80	DEVDSPLY	DISPLAY FIELDS
488	(1E8)	ADDRESS	4	DEVACPTN	ACTIVE 3270 PTN CONTROL BLOCK ADDRESS
488	(1E8)	ADDRESS	4	DEVFTBAD	ADDR OF LU7 FORMAT TABLE
500	(1F4)	ADDRESS	4	DEVPTNAD	3270 PTN CONTROL BLOCK ADDRESS, FIRST IN CHAIN
521	(209)	BITSTRING	1	DEVFEAT2	3270 DEVICE FEATURE BITS
		1...		DEVDBCS	DBCS SUPPORT
		.1..		DEVFLDOL	FIELD OUTLINING SUPPORT
522	(20A)	ADDRESS	1	DEVAID	AID BYTE FOR 3270
528	(210)	UNSIGNED	1	DEVPSNO	NUMBER OF PS'S SUPPORTED
536	(218)	ADDRESS	4	DEVCSIDA	ADDR OF CHAR SET ID VALUES
548	(224)	BITSTRING	4	DEV3270F	3270 FLAGS
549	(225)	...1		DEVINIHB	INPUT INHIBITED
550	(226)	...1		DEVPTNST	DEVICE IN PARTITIONED STATE
551	(227)	.1..		DEVUOM	UNIT OF MEASURE FOR SCREEN SIZE, 0-INCH, 1-MM
551	(227)	..1.		DEVVARCC	VARIABLE CHARACTER CELL SIZE
552	(228)	ADDRESS	4	DEVATRTB	ATTRIBUTE TABLE ADDR
556	(22C)	SIGNED	2	DEVATRCT	ATTRIBUTE COUNT FIELD

Offsets					
Dec	Hex	Type	Len	Name (Dim)	Description
SNA FIELDS					
806	(326)	CHARACTER	26	DEVBNDRU	BYTES 1-26 OF BIND COMMAND
806	(326)	BITSTRING	1	DEVBND01	BYTE 1 OF BIND COMMAND
		1111		DEVBFORM	BIND FORMAT
		 1111		DEVBTYPE	BIND TYPE
807	(327)	UNSIGNED	1	DEVBNDFM	FUNCTION MANAGER PROFILE
808	(328)	UNSIGNED	1	DEVBNDTS	TRANSMISSION SUB-SYSTEM PROFILE
809	(329)	BITSTRING	1	DEVBPRIP	PRIMARY PROTOCOLS
		11..		*	NOT CHECKED BY WSim
		..1.		DEVBPDEF	PRIMARY MAY ASK FOR DEF RESP
		...1		DEVBPEXC	PRIMARY MAY ASK FOR EXC RESP
		.... 111.		*	NOT CHECKED BY WSim
		1		DEVBPSEB	PRIMARY MAY SEND END BRACKET
810	(32A)	BITSTRING	1	DEVBSECP	SECONDARY PROTOCOLS
		11..		*	NOT CHECKED BY WSim
		..1.		DEVBSDEF	SECONDARY MAY ASK FOR DEF RESP
		...1		DEVBSEXC	SECONDARY MAY ASK FOR EXC RESP
		.... 111.		*	NOT CHECKED BY WSim
		1		DEVBSSEB	SECONDARY MAY SEND END BRACKET
811	(32B)	BITSTRING	2	DEVBCOMP	COMMON PROTOCOLS
		1...		DEVBNSEG	NO SEGMENTING SUPPORT
		.1..		DEVBFMH	FM HEADERS WILL BE USED
		..1.		DEVBRACK	BRACKETS WILL BE USED
		...1		DEVBNDBT	BRACKET TERMINATION
		 1111		*	NOT CHECKED BY WSim
		11..		DEVBMODE	MODE SELECTION
		1...		DEVBNDFF	FLIP-FLOP MODE
		.1..		DEVBNDCN	CONTENTION MODE
		..1.		DEVBRCOV	SENDER RESPONSIBLE FOR RECOVERY
		...1		DEVBFSP	PRIMARY IS FIRST SPEAKER
		.... 111.		*	NOT CHECKED BY WSim
		1		DEVBCONR	PRIMARY WINS CONTENTION
813	(32D)	UNSIGNED	1	DEVBSSPC	SECONDARY SEND PACING COUNT
		1...		DEVBSSTG	SEC STAGING INFO, 0=1 STAGE
814	(32E)	UNSIGNED	1	DEVBSRPC	SECONDARY RECEIVE PACING COUNT
		1...		DEVBASPI	ADAPTIVE SESSION PACING INFO
815	(32F)	UNSIGNED	1	DEVBSMRU	SECONDARY SEND RU SIZE--SPECIAL FORMAT
816	(330)	UNSIGNED	1	DEVBPMRU	PRIMARY SEND RU SIZE--SPECIAL FORMAT
817	(331)	UNSIGNED	1	DEVBPSPC	PRIMARY SEND PACING COUNT
		1...		DEVBPSTG	PRI STAGING INFO, 1=1 STAGE
818	(332)	UNSIGNED	1	DEVBPRPC	PRIMARY RECEIVE PACING COUNT
819	(333)	UNSIGNED	1	DEVLUTYP	LOGICAL UNIT TYPE
820	(334)	CHARACTER	1	DEVBND15	BYTE 15 OF BIND COMMAND
821	(335)	CHARACTER	1	DEVBND16	BYTE 16 OF BIND COMMAND
822	(336)	CHARACTER	1	DEVBND17	BYTE 17 OF BIND COMMAND
823	(337)	CHARACTER	1	DEVBND18	BYTE 18 OF BIND COMMAND
824	(338)	CHARACTER	1	DEVBND19	BYTE 19 OF BIND COMMAND
825	(339)	CHARACTER	1	DEVBND20	BYTE 20 OF BIND COMMAND
826	(33A)	CHARACTER	1	DEVBND21	BYTE 21 OF BIND COMMAND
827	(33B)	CHARACTER	1	DEVBND22	BYTE 22 OF BIND COMMAND
828	(33C)	CHARACTER	1	DEVBND23	BYTE 23 OF BIND COMMAND
829	(33D)	CHARACTER	1	DEVBND24	BYTE 24 OF BIND COMMAND
830	(33E)	CHARACTER	1	DEVBND25	BYTE 25 OF BIND COMMAND
831	(33F)	CHARACTER	1	DEVBND26	BYTE 26 OF BIND COMMAND
831	(33F)	BITSTRING	1	DEVBCRYO	CRYPTOGRAPHIC OPTIONS
		11..		DEVBCRYP	PRIVATE CRYPTOGRAPHY
		..11		DEVBCRYS	SESSION LEVEL
		 1111		DEVBCRYL	LENGTH OF CRYPT FIELDS
CHARACTER SETS IDENTIFICATIONS					
0	(0)	STRUCTURE	16	DEVCSIDN	CHAR SET ID NODE
0	(0)	ADDRESS	4	DEVNXTCS	NEXT CHAR SET ID PTR
4	(4)	CHARACTER	4	DEVBCSID	BASE CHAR SET ID
8	(4)	CHARACTER	4	DEVACSID	APL CHAR SET ID
12	(C)	CHARACTER	4	DEVDCSID	DBCS CHAR SET ID

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
DEVICE SEQUENCE AND INDEX COUNTERS CONTROL BLOCK					
0	(0)	STRUCTURE	*	DEVCTRS	SEQ AND INDX CNTRS
0	(0)	UNSIGNED	4	DEVSEQ	DEVICE SEQUENCE COUNTER
4	(4)	UNSIGNED	4	DEVSEQCT (*)	DEVICE INDEX COUNTERS

Constants

Len	Type	Value	Name	Description
DEVICE TYPES				
1	HEX	91	DEVFTP	FTP DEVICE
1	HEX	92	DEVFTPD	FTP DATA CONNECTION
1	HEX	93	DEVSTCP	SIMPLE TCP DEVICE
1	HEX	94	DEV32TN	TELNET 3270 DEVICE
1	HEX	95	DEVTNE	TELNET 3270E DEVICE
1	HEX	96	DEVTNEP	TELNET 3270E PRINTER
1	HEX	97	DEVSUDP	SIMPLE UDP
1	HEX	98	DEVLNMD	TELNET LINE MODE NVT DEVICE
1	HEX	99	DEV5250	TELNET 5250 DEVICE
1	HEX	E0	DEVLU0	LU TYPE 0
1	HEX	E1	DEVLU1	LU TYPE 1
1	HEX	E2	DEVLU2	LU TYPE 2
1	HEX	E3	DEVLU3	LU TYPE 3
1	HEX	E4	DEVLU4	LU TYPE 4
1	HEX	E5	DEVLU5	LU TYPE 5
1	HEX	E6	DEVLU6	LU TYPE 6
1	HEX	E7	DEVLU7	LU TYPE 7
1	HEX	E9	DEVLU62	LU TYPE 6.2
1	HEX	EA	DEVAPPC	APPC TP
1	HEX	FF	DEVDUMMY	DUMMY DEV FOR USER INTERFACE

Cross-Reference

Name	Hex Offset	Hex Value
DEV	0	
DEVACPTN	1E8	
DEVACSID	8	
DEVAID	20A	
DEVATRCT	22C	
DEVATRTB	228	
DEVBASPI	32E	80
DEVBCOMP	32B	
DEVBCONR	32B	01
DEVBCRYL	33F	08
DEVBCRYO	33F	
DEVBCRYP	33F	80
DEVBCRYS	33F	20
DEVBCSID	4	
DEVBFMH	32B	40
DEVBFORM	326	80
DEVBFSP	32B	10
DEVBMODE	32B	80
DEVBNDBT	32B	10
DEVBNDCN	32B	40
DEVBNDFF	32B	80
DEVBNDFM	327	
DEVBNDRU	326	
DEVBNDTS	328	
DEVBND01	326	
DEVBND15	334	
DEVBND16	335	
DEVBND17	336	
DEVBND18	337	
DEVBND19	338	
DEVBND20	339	
DEVBND21	33A	
DEVBND22	33B	
DEVBND23	33C	
DEVBND24	33D	
DEVBND25	33E	
DEVBND26	33F	
DEVBNSEG	32B	80
DEVBPDEF	329	20
DEVBPEXC	329	10
DEVBPMRU	330	
DEVBPRIP	329	
DEVBPRPC	332	
DEVBPSEB	329	01

Name	Hex Offset	Hex Value
DEVBPSPC	331	
DEVBPSTG	331	80
DEVBRACK	32B	20
DEVBRCOV	32B	20
DEVBSDEF	32A	20
DEVBSECP	32A	
DEVBSEXC	32A	10
DEVBSMRU	32F	
DEVBSRPC	32E	
DEVBSSEB	32A	01
DEVBSSPC	32D	
DEVBSSTG	32D	80
DEVBTYPE	326	08
DEVCSIDA	218	
DEVCSIDN	0	
DEVCTRAD	40	
DEVCTRS	0	
DEVCURSR	24	
DEVDBCS	209	80
DEVDEV	179	80
DEVDCSID	C	
DEVDSPLY	1E8	
DEVEWAIT	20	01
DEVFEAT2	209	
DEVFLAG1	20	
DEVFLAG4	22	
DEVFLDOL	209	40
DEVFTBAD	1E8	
DEVIBUFL	17A	
DEVINBUF	14	
DEVINIHB	225	10
DEVLINAD	FC	
DEVLUTYP	333	
DEVNAME	34	
DEVNCBAD	E4	
DEVNXDEV	100	
DEVNXTCS	0	
DEVNXTRM	1B0	
DEVOBUFL	1C	
DEVOTBUF	18	
DEVPSNO	210	
DEVPTNAD	1F4	
DEVPTNST	226	10
DEVQUIEC	22	20
DEVSAVAD	C8	
DEVSEQ	0	
DEVSEQCT	4	
DEVSNA	179	40
DEVSWCH	44	
DEVTRMAD	104	
DEVTYPE	179	
DEVUOM	227	40
DEVUSER	55	
DEVUSRAD	1AC	
DEVUSRLN	19E	
DEVVARCC	227	20
DEVWAIT	20	04

Log Display Record Header Format (LDS)

Size in bytes:	16 (X'10')
Function:	Defines the format of the header for log records that contain information about a display screen image. An LDS occurs in the data portion of a log record after the normal 88 byte log record header.

LOG DISPLAY RECORD HEADER

Offsets **Dec**	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	16	LDS	LOG DISPLAY HEADER
0	(0)	SIGNED	4	LDSTOTLN	TOTAL LENGTH OF HEADER AND BUFFERS
4	(4)	BITSTRING	1	LDSAID	DEVAID AT TIME OF LOG
5	(5)	SIGNED	2	LDS7CUR	CURSOR POSITION FOR LU7
5	(5)	BITSTRING	1	LDSFEAT	DEV FEATURE BITS MAPPED INTO LDS
		1...		LDSFECS	EXTENDED CHARACTER SET FEATURE
		.1..		LDSAAPL	APL ALTERNATE CHARACTER SET FEATURE
		..1.		LDSFEXTF	EXTENDED FUNCTION SUPPORT
		...1		LDSCOLR	COLOR SUPPORT
		.... 1...		LDSFHL	HIGHLIGHTING SUPPORT
		.... .1..		LDSFUBFP	UNBUFFERED PRINTER ATTACHED TO 3277, 3277S
		.... ..1.		LDSFFLDV	FIELD VALIDATION SUPPORT
		1		LDSORANG	NO COLOR, 0=GREEN, 1=ORANGE
6	(6)	BITSTRING	1	LDSFLAG	FLAGS
		1...		LDSPTNST	DEVICE IS IN PARTITIONED STATE
		.1..		LDSDBCS	DBCS SUPPORT
		..1.		LDSFLDOL	FIELD OUTLINING SUPPORT
7	(7)	BITSTRING	1	LDSFLAG1	FLAGS2
		11..		LDSCOM	INDICATE WHERE DISPLAY IS LOGGED
		..11		LDSPRFMT	INDICATE PRINTER FORMAT
		..1.		LDSPFMT1	PRINT FORMAT INDICATOR
		...1		LDSPFMT2	PRINT FORMAT INDICATOR
		.... 1...		LDS327P	INDICATE 3270 PRINTER
8	(8)	SIGNED	2	LDSUABSZ	USABLE AREA BUFFER SIZE WHEN LDSPTNST=ON
8	(8)	SIGNED	2	LDS7BSZ	BUFFER SIZE FOR LU7
10	(A)	UNSIGNED	1	LDSUARSZ	USABLE AREA ROW SIZE WHEN LDSPTNST=ON
10	(A)	UNSIGNED	1	LDS7RSZ	ROW SIZE FOR LU7
11	(B)	BITSTRING	1	LDS7ST	STATE VECTOR FOR LU7
12	(C)	SIGNED	2	LDSMAXAT	MAXIMUM ATTRIBUTES IN A PTN
14	(E)	CHARACTER	2	LDSRSV1	RESERVED
16	(10)	CHARACTER		LDSDATAX	PTN, DISPLAY BUFFER, ATTRIBUTE TABLE, EAB, FVB

Constants

Len	**Type**	**Value**	**Name**	**Description**
LDSCOM DEFINITIONS				
0	BIT	11	LDSLOGDS	DSPY LOG VIA LOG DISPLAY CMND
0	BIT	01	LDSBMSG	DSPY LOG AT BEGIN OF MSG GEN
0	BIT	10	LDSEMSG	DSPY LOG AT END OF MSG GEN
0	BIT	00	LDSSTPRT	DSPY LOG VIA START PRINT

Cross-Reference

Name	Hex Offset	Hex Value
LDS	0	
LDSAAPL	5	40
LDSAID	4	
LDSCOLR	5	10
LDSCOM	7	80
LDSDATAX	10	
LDSDBCS	6	40
LDSFEAT	5	
LDSFECS	5	80
LDSFEXTF	5	20
LDSFFLDV	5	02
LDSFHL	5	08
LDSFLAG	6	
LDSFLAG1	7	
LDSFLDOL	6	20
LDSFUBFP	5	04
LDSMAXAT	C	
LDSORANG	5	01
LDSPFMT1	7	20
LDSPFMT2	7	10
LDSPRFMT	7	20
LDSPTNST	6	80
LDSRSV1	E	
LDSTOTLN	0	
LDSUABSZ	8	
LDSUARSZ	A	
LDS327P	7	08
LDS7BSZ	8	
LDS7CUR	5	
LDS7RSZ	A	
LDS7ST	B	

Line Control Block (LIN)

Size in bytes:	192 (X'C0')
Pointed to by:	Chain based on NCBLINAD, DEVLINAD, TRMLINAD, PULINAD
Function:	Line control block structure.

LINE CONTROL BLOCK

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
12	(0C)	ADDRESS	8	LINNCBAD	NCB ADDRESS
24	(18)	CHARACTER	8	LINNAME	LINE NAME
32	(20)	CHARACTER	6	LINID	EBCDIC LINE NUMBER
48	(30)	ADDRESS	4	LINNXLIN	ADDRESS OF NEXT LIN (NCB)
52	(34)	ADDRESS	4	LINNXLNK	ADDRESS OF NEXT LIN (CNTLR)
91	(51)	BITSTRING	1	LINFLAG1	FLAG FIELDS
		.... 1...		LINSTRTD	LINE STARTED
120	(78)	ADDRESS	4	LINCTRAD	LINE COUNTERS ADDRESS
192	(C0)	ADDRESS	4	LINTRTX	TERMINAL RESOLUTION TABLE
LINE SEQUENCE AND INDEX COUNTERS CONTROL BLOCK					
0	(0)	STRUCTURE	*	LINCTRS	SEQ AND INDX CNTRS
0	(0)	UNSIGNED	4	LINSEQ	LINE SEQUENCE COUNTER
4	(4)	UNSIGNED	4	LINSEQCT (*)	LINE INDEX COUNTERS

Cross-Reference

Name	Hex Offset	Hex Value
LINCTRAD	78	
LINID	20	
LINNAME	18	
LINNCBAD	0C	
LINNXLIN	30	
LINNXLNK	34	
LINSEQ	0	
LINSEQCT	4	
LINSTRTD	51	01
LINTRTX	C0	

Log Record Header Format (LOG)

Size in bytes:	88 (X'58')
Function:	Defines the format of the header for all records written to the WSim log tape.

Offsets Dec	 Hex	 Type	 Len	 Name (Dim)	 Description
0	(0)	STRUCTURE	88	LOG	LOG RECORD HEADER FORMAT
0	(0)	CHARACTER	4	LOGHEADR	RECORD HEADER
0	(0)	SIGNED	2	LOGLRECL	LOGICAL RECORD LENGTH
2	(2)	ADDRESS	1	LOGRECTP	SPANNED RECORD TYPE
		1111 11..		*	UNREFERENCED BITS - ALWAYS ZERO
		.... ..1.		LOGNFSPN	NOT FIRST SPAN - MIDDLE OR LAST
		1		LOGNLSPN	NOT LAST SPAN - FIRST OR MIDDLE
3	(3)	CHARACTER	1	LOGRSV01	RESERVED
4	(4)	CHARACTER	8	LOGNTNAM	NETWORK NAME
12	(C)	CHARACTER	8	LOGLNNAM	LINE NAME
20	(14)	CHARACTER	8	LOGSBNAM	SUBAREA NAME
28	(1C)	CHARACTER	8	LOGTRMID	TERMINAL NAME
36	(24)	CHARACTER	8	LOGDECK	MESSAGE DECK NAME
44	(2C)	ADDRESS	1	LOGTRMTP	TERMINAL TYPE
		1...		LOGDEV	TERMINAL IS A DEVICE
		.1..		LOGSNA	SNA TERMINAL OR DEVICE
45	(2D)	UNSIGNED	3	LOGLINID	HEX LINE ID
48	(30)	CHARACTER	12	LOGTIME	TIMESTAMP FIELDS
48	(30)	CHARACTER	4	LOGSTART	START TIMESTAMP
52	(34)	CHARACTER	4	LOGSTOP	STOP TIMESTAMP
52	(34)	CHARACTER	4	LOGDATE	CURRENT DATE
56	(38)	CHARACTER	4	LOGRDYTM	READY TIMESTAMP
56	(38)	CHARACTER	2	LOGLEVEL	CURRENT WSim RELEASE AND VERSION
56	(38)	CHARACTER	1	LOGVERSN	VERSION
57	(39)	CHARACTER	1	LOGRLEAS	RELEASE
60	(3C)	SIGNED	2	LOGLENG	LENGTH OF DATA
62	(3E)	UNSIGNED	2	LOGSEQNO	RECORD SEQUENCE NUMBER
64	(40)	BITSTRING	2	LOGFLAG	RECORD TYPE FLAGS
		1...		LOGMSG	MESSAGE DATA RECORD
		.1..		LOGINFO	INFORMATIONAL RECORD
		..1.		LOGTRACE	LINE OR CPI-C TRACE DATA
		...1		LOGMARKR	MARKER RECORD
		.... 1...		LOGCNSLE	CONSOLE COMMAND
		.... .1..		LOGMDR	MDR RECORD
		.... ..1.		LOGLOGDS	LOG DISPLAY BUFFERS RECORD
		1		LOGMSGTR	MSG TRACE TYPE RECORDS
		1...		LOGLOGRC	LOG TYPE RECORD
		.1..		LOGVRFY	VERIFY RECORD
		..1.		LOG21LOG	PU21 RESOURCE IS BEING LOGGED
		...1		LOGCXID	PU21 CHANNEL XID
		.... 1...		LOG21CHN	PU21 CHANNEL DATA
		.... .1..		LOGSTLTR	STL TRACE RECORD
		.... ..1.		LOGNFSGM	NOT FIRST LOG SEGMENT
		1		LOGNLSGM	NOT LAST LOG SEGMENT

Offsets					
Dec	Hex	Type	Len	Name (Dim)	Description
66	(42)	BITSTRING	3	LOGFLAGM	MODIFIER FLAGS
		1...		LOGTRANS	TRANSMIT RECORD
		.1..		LOGLDATA	LOST DATA IN LAST RECORD
		..1.		LOGEOT	END OF TRANSMISSION (EOT)
		...1		LOGCONV	CONVERSATIONAL REPLY
		...1		LOGCPICV	CPI-C TRACE WITH ON=VERB OFF=MSG
		...1		LOGFMH5	CPI-C FMH-5 MESSAGE DATA
		.... 1...		LOGTYPE3	TYPE 3 LINE TRACE
		.... 1...		LOGVAPPL	VTAMAPPL (VTAM API) MESSAGE DATA
		.... .1..		LOGHEX	HEXADECIMAL DATA
		.... .1..		LOGAPCLU	APPCLU CPI-C TRACE OR MSG DATA
		.... ..1.		LOGDISC	DISCONNECT (DLE EOT)
		1		LOGNAM	NAME FIELD PRECEDES DATA
		1		LOGLHDRS	LAN HEADERS INCLUDED IN DATA
		1...		LOGOPSCR	CONSOLE COMMAND FROM SCRIPT
		1...		LOGLAN	LAN DATA
		.1..		LOGHDRS	SNA HEADERS ONLY
		..1.		LOGCHN	CHANNEL TRANSFER
		...1		LOGGROUP	MSG TO BE GROUPED
		.... 1...		LOGCDLOG	CDLOG LOGGING OPTION
		.... .1..		LOGXMTLG	LINE TRACE FOR FDX XMIT LEG
		.... .1..		LOGENET	ETHERNET LAN
		.... ..1.		LOGLGCB	LOG STMT CONTROL BLOCK DATA
		.... ..1.		LOGENV2	ETHERNET V2 (ON) OR 802.3 (OFF)
		1		LOGCDSIM	CROSS-DOMAIN MESSAGE DATA
		1...		LOGSDLC	SDLC FIELDS EXIST
		.1..		LOGFSTRC	FIRST RECORD OF A GROUP OF LOG DISPLAY RECORDS
		.1..		LOGM128	SDLC MODULO 128 FOR SDLC DATA RECORDS
		.1..		LOGCPICM	CPI-C TRACE COMPRESSED MESSAGE
		.1..		LOGCPILD	CPI-C LOG DATA MESSAGE DATA
		..1.		LOGLSTRC	LAST RECORD OF A GROUP OF LOG DISPLAY RECORDS
		..1.		LOGVTAMI	PROCESSED/GENERATED BY VTAM INTERFACE ROUTINES
		..1.		LOGCPIHD	CPI-C TRACE MESSAGE WITH HEX DATA
		..1.		LOGCPIVD	CPI-C TRACE VERB DATA
		...1		LOGXLOG	X.25 DATA
		.... 1...		LOGPSH	X.25 DATA CONTAINS PSH
		.... 1...		LOGEN2CL	ETHERNET V2 CONNECTIONLESS PROTOCOL
		.... .1..		LOG3725T	3725 TRACE DATA
		.... .1..		LOGPAD	X.25 TWX PAD DATA
		.... .1..		LOGASCII	DATA IS IN ASCII
		.... ..1.		LOGSITTR	SCANNER INTERNAL TRACE DATA
		.... ..1.		LOGHUCMD	HDLC UNNUMBERED COMMAND FRAME
		.... ..1.		LOGTLNET	TCPIP 3270 LOG RECORD
		1		LOGQLLC	X.25 QLLC DATA
		1		LOGFRRLY	FRAME RELAY RECORD
69	(45)	CHARACTER	1	LOGUSER	USER DATA BYTE
70	(46)	CHARACTER	5	LOGSESNO	LOGICAL UNIT SESSION NUMBER
70	(46)	CHARACTER	1	*	NOT USED
71	(47)	CHARACTER	4	LOGLCHNO	LCH NUMBER FOR DYNAMIC LCH
75	(4B)	CHARACTER	1	LOGRSV02	RESERVED
76	(4C)	CHARACTER	4	LOGDLC	SDLC BYTES FOR SDLC DATA RECORD
76	(4C)	UNSIGNED	2	LOGDSPSQ	COUNT FOR DISPLAY RECORDS
80	(50)	CHARACTER	8	LOGTIMEB	TIME BINARY WHEN RECORD LOGGED
80	(50)	SIGNED	4	LOGTODB	TIME OF DAY IN BINARY 1/100 SEC
84	(54)	SIGNED	4	LOGDATEB	DATE WORD IN 00YYDDDF FORMAT
88	(58)	CHARACTER		LOGDATAX	START OF LOG DATA AREA

Constants

SPANNED RECORD TYPES

Len	Type	Value	Name	Description
1	HEX	00	LOGCMPLT	COMPLETE RECORD
1	HEX	01	LOGFSTSG	FIRST SPANNED SEGMENT
1	HEX	02	LOGLSTSG	LAST SPANNED SEGMENT
LOG TERMINAL TYPES				
1	HEX	30	LOG327T	TCPIP 3270
LOG DEVICE TYPES				
1	HEX	91	LOGFTPCC	FTP COMMAND
1	HEX	92	LOGFTPDC	FTP DATA
1	HEX	93	LOGSTCP	SIMPLE TCP
1	HEX	94	LOG32TN	TELNET 3270 DEVICE
1	HEX	95	LOG32TE	TELNET 3270E DEVICE
1	HEX	96	LOG32TP	TELNET 3270E PRINTER
1	HEX	97	LOGSUDP	SIMPLE UDP
1	HEX	98	LOGLNMD	TELNET LINE MODE NVT DEVICE
1	HEX	99	LOG5250	TELNET 5250 DEVICE
1	HEX	E0	LOGLU0	LU TYPE 0
1	HEX	E1	LOGLU1	LU TYPE 1
1	HEX	E2	LOGLU2	LU TYPE 2
1	HEX	E3	LOGLU3	LU TYPE 3
1	HEX	E4	LOGLU4	LU TYPE 4
1	HEX	E5	LOGLU5	LU TYPE 5
1	HEX	E6	LOGLU6	LU TYPE 6
1	HEX	E7	LOGLU7	LU TYPE 7
1	HEX	E9	LOGLU62	LU TYPE 6.2
1	HEX	EA	LOGAPPC	APPC TP

Cross-Reference

Name	Hex Offset	Hex Value
LOG	0	
LOGAPCLU	42	04
LOGASCII	44	04
LOGCDLOG	43	08
LOGCDSIM	43	01
LOGCHN	43	20
LOGCNSLE	40	08
LOGCONV	42	10
LOGCPICM	44	40
LOGCPICV	42	10
LOGCPIHD	44	20
LOGCPILD	44	40
LOGCPIVD	44	20
LOGCXID	41	10
LOGDATAX	58	
LOGDATE	34	
LOGDATEB	54	
LOGDECK	24	
LOGDEV	2C	80
LOGDISC	42	02
LOGDLC	4C	
LOGDSPSQ	4C	
LOGENET	43	04
LOGENV2	43	02
LOGEN2CL	44	08
LOGEOT	42	20
LOGFLAG	40	
LOGFLAGM	42	
LOGFMH5	42	10
LOGFRRLY	44	01
LOGFSTRC	44	40
LOGGROUP	43	10
LOGHDRS	43	40
LOGHEADR	0	
LOGHEX	42	04
LOGHUCMD	44	02
LOGINFO	40	40
LOGLAN	43	80
LOGLCHNO	47	
LOGLDATA	42	40
LOGLENG	3C	
LOGLEVEL	38	
LOGLGCB	43	02
LOGLHDRS	42	01
LOGLINID	2D	
LOGLNNAM	C	
LOGLOGDS	40	02
LOGLOGRC	41	80
LOGLRECL	0	
LOGLSTRC	44	20
LOGMARKR	40	10
LOGMDR	40	04
LOGMSG	40	80
LOGMSGTR	40	01
LOGM128	44	40
LOGNAM	42	01
LOGNFSGM	41	02
LOGNFSPN	2	02
LOGNLSGM	41	01
LOGNLSPN	2	01
LOGNTNAM	4	
LOGOPSCR	43	80
LOGPAD	44	04
LOGPSH	44	08
LOGQLLC	44	01
LOGRDYTM	38	
LOGRECTP	2	
LOGRLEAS	39	
LOGRSV01	3	
LOGRSV02	4B	
LOGSBNAM	14	
LOGSDLC	44	80
LOGSEQNO	3E	
LOGSESNO	46	
LOGSITTR	44	02
LOGSNA	2C	40
LOGSTART	30	
LOGSTLTR	41	04
LOGSTOP	34	
LOGTIME	30	
LOGTIMEB	50	
LOGTLNET	44	02
LOGTODB	50	
LOGTRACE	40	20
LOGTRANS	42	80
LOGTRMID	1C	
LOGTRMTP	2C	
LOGTYPE3	42	08
LOGUSER	45	
LOGVAPPL	42	08
LOGVERSN	38	
LOGVRFY	41	40
LOGVTAMI	44	20
LOGXLOG	44	10
LOGXMTLG	43	04
LOG21CHN	41	08
LOG21LOG	41	20
LOG3725T	44	04

Network Control Block (NCB)

Size in bytes:	680 (X'2A8')
Pointed to by:	Chain based on TVTNCBAD TRMNCBAD, DEVNCBAD, LINNCBAD
Function:	The NCB is the primary control block in a WSim network simulation. It contains information that applies to all resources defined in a network. One NCB is built for each network simulated by WSim.

NETWORK CONTROL BLOCK

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	664	NCB	NETWORK CONTROL BLOCK
0	(0)	SIGNED	2	NCBLENG	LENGTH OF NCB
2	(2)	BITSTRING	1	NCBFLAG1	FIRST FLAG BYTE
		1...		NCBSTRTD	NETWORK ACTIVE
12	(C)	CHARACTER	8	NCBNAME	CONTROL BLOCK NAME
20	(14)	CHARACTER	24	NCBHEADR	INTERVAL REPORT HEADER
44	(2C)	ADDRESS	4	NCBNXNCB	NCB CHAIN LINK FIELD
VARIOUS VALUES AND POINTERS					
100	(64)	UNSIGNED	4	NCBSWCH	32 NETWORK SWITCHES
104	(68)	ADDRESS	4	NCBLINAD	ADDR OF FIRST LIN FOR NETWORK
108	(6C)	ADDRESS	4	NCBUSER	ADDR OF NETWORK WIDE USERAREA
MISCELLANEOUS COUNTERS AND VALUES					
172	(AC)	SIGNED	2	NCBUSRLN	LENGTH OF NETWORK USER AREA
NCB DQE & TQE					
280	(118)	ADDRESS	4	NCBCTRAD	NETWORK COUNTERS ADDRESS
284	(11C)	UNSIGNED	1	NCBCNTRS	MINIMUM NUMBER OF INDEX COUNTERS
285	(11D)	UNSIGNED	1	NCBCNTRO	NUMBER CODED ON CNTRS= OPERAND
286	(11E)	SIGNED	2	NCBCTRLN	LENGTH OF STORAGE FOR COUNTERS
SEQUENCE AND INDEX COUNTER BLOCK					
0	(0)	STRUCTURE	*	NCBCTRS	SEQUENCE AND INDEX COUNTERS
0	(0)	UNSIGNED	4	NCBSEQ	NETWORK SEQUENCE COUNTER
4	(4)	UNSIGNED	4	NCBSEQCT (*)	NETWORK INDEX COUNTERS

Cross-Reference

Name	Hex Offset	Hex Value
NCB	0	
NCBCNTRO	11D	
NCBCNTRS	11C	
NCBCTRAD	118	
NCBCTRLN	11E	
NCBCTRS	0	
NCBFLAG1	2	
NCBHEADR	14	
NCBLENG	0	
NCBLINAD	68	
NCBNAME	C	
NCBNXNCB	2C	
NCBSEQ	0	
NCBSEQCT	4	

Name	Hex Offset	Hex Value
NCBSTRTD	2	80
NCBSWCH	64	
NCBUSER	6C	
NCBUSRLN	AC	

Print Control Block (PRT)

Size in bytes:	172 (X'AC')
Function:	Contains an output line to be printed by ITPPUT and maintains information about the status of the output page.

PRINT CONTROL BLOCK

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	48	PRT	PRT CONTROL BLOCK
0	(0)	ADDRESS	4	PRTDCBAD	ADDR OF PRINT DCB
4	(4)	UNSIGNED	1	PRTLNCNT	CURRENT LINE COUNT ON PAGE
5	(5)	UNSIGNED	1	PRTHDRNO	NUMBER OF HEADER LINES
6	(6)	UNSIGNED	2	PRTPACNT	CURRENT PAGE COUNT
8	(8)	UNSIGNED	1	PRTENDPG	MAXIMUM NUMBER OF LINES ON PAGE
9	(9)	BITSTRING	1	PRTFLG	FLAGS
		1...		PRTUPCAS	UPPER CASE ON LOGLIST
		.1..		PRTAMD31	PRINT TASK RUNNING IN 31 BIT ADDRESSING MODE
		..1.		PRTSPHDR	SUPPRESS HEADER
		...1		PRTOPEN	REQUEST TO OPEN PRINT DATA SET
		.... 1...		PRTCLOSE	REQUEST TO CLOSE PRINT DATA SET
		.... .1..		PRTOPEND	PRINT DATA SET IS OPEN
		1		PRTHILIT	PRINT LINE HIGHLIGHTED
10	(A)	CHARACTER	27	PRTTITLE	TOP OF PAGE TITLE
37	(25)	CHARACTER	1	PRTFLG2	MORE FLAGS
		1...		PRTOPTJ	OPTCD=J CODED, TRC INCLUDED
		..1.		PRTRFMVB	VARIABLE LENGTH RECORDS
		...1		PRTSOSI1	PRMODE SOSI1
		.... 1...		PRTSOSI2	PRMODE SOSI2
		.... .1..		PRTASCII	ASCII DATA
38	(26)	SIGNED	2	PRTDATLN	LENGTH OF DATA AREA
40	(28)	ADDRESS	4	PRTDATAD	ADDRESS OF DATA AREA
44	(2C)	ADDRESS	4	PRTBUFAD	ADDRESS OF CURRENT PRINT BUFFER
48	(30)	CHARACTER	*	PRTHDRS	HEADERS
0	(0)	STRUCTURE	256	PRTPRTLX	EXTENDED PRINT LINE
0	(0)	CHARACTER	1	PRTCCX	CARRIAGE CONTROL FIELD
1	(1)	CHARACTER	255	PRTDATAX	MAXIMUM DATA AREA
0	(0)	STRUCTURE	133	PRTPRTLN	STANDARD PRINT LINE
0	(0)	CHARACTER	1	PRTCC	CARRIAGE CONTROL FIELD
1	(1)	CHARACTER	132	PRTDATA	STANDARD DATA AREA
0	(0)	STRUCTURE	4	PRTVBHDR	VARIABLE RECORD HEADER
0	(0)	SIGNED	2	PRTRECLN	LENGTH OF RECORD
2	(2)	BITSTRING	2	PRTSEGMT	SEGMENTING BITS
4	(4)	CHARACTER	*	PRTVDATA	VARIABLE DATA

Cross-Reference

Name	Hex Offset	Hex Value
PRT	0	
PRTAMD31	9	40
PRTASCII	25	04
PRTBUFAD	2C	
PRTCC	0	
PRTCCX	0	
PRTCLOSE	9	08
PRTDATA	1	
PRTDATAD	28	
PRTDATAX	1	

Name	Hex Offset	Hex Value
PRTDATLN	26	
PRTDCBAD	0	
PRTENDPG	8	
PRTFLG	9	
PRTFLG2	25	
PRTHDRNO	5	
PRTHDRS	30	
PRTHILIT	9	01
PRTLNCNT	4	
PRTOPEN	9	10
PRTOPEND	9	04
PRTOPTJ	25	80
PRTPACNT	6	
PRTPRTLN	0	
PRTPRTLX	0	
PRTRECLN	0	
PRTRFMVB	25	20
PRTSEGMT	2	
PRTSOSI1	25	10
PRTSOSI2	25	08
PRTSPHDR	9	20
PRTTITLE	A	
PRTUPCAS	9	80
PRTVBHDR	0	
PRTVDATA	4	

Display Partition Control Block (PTN)

Size in bytes:	60 (X'3C')
Pointed to by:	Chain based on DEVPTNAD
Function:	Contains information about a screen partition for a 3270 display or printer device.

PARTITION CONTROL BLOCK FOR 3270 DEVICES

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	60	PTN	PARTITION CONTROL BLOCK
0	(0)	ADDRESS	4	PTNATRTB	ATTRIBUTE TABLE ADDRESS
4	(4)	SIGNED	2	PTNATRCT	ATTRIBUTE COUNT
6	(6)	SIGNED	2	PTNPSSIZ	PRESENTATION SPACE BUFFER SIZE
8	(8)	ADDRESS	4	PTNPSBUF	PRESENTATION SPACE BUFFER ADDRESS
12	(C)	ADDRESS	4	PTNEABUF	EXTENDED ATTRIBUTE BUFFER ADDRESS
16	(10)	ADDRESS	4	PTNFVBUF	FIELD VALIDATION BUFFER ADDRESS
20	(14)	SIGNED	2	PTNATRSZ	ATTRIBUTE TABLE SIZE
22	(16)	SIGNED	2	PTNCCP	CURRENT CURSOR POSITION STARTING WITH ONE
24	(18)	ADDRESS	4	PTNNXPTN	NEXT PTN ADDRESS IN CHAIN, ZERO IF LAST
28	(1C)	CHARACTER	24	PTNPARM	PARTITION SIZE PARAMETERS
28	(1C)	SIGNED	2	PTND	DEPTH OF PRESENTATION SPACE IN ROWS
30	(1E)	SIGNED	2	PTNW	WIDTH OF PRESENTATION SPACE IN COLUMNS
32	(20)	SIGNED	2	PTNRV	ROW OFFSET FOR VIEWPORT ON USABLE AREA
34	(22)	SIGNED	2	PTNCV	COLUMN OFFSET FOR VIEWPORT ON USABLE AREA
36	(24)	SIGNED	2	PTNDV	DEPTH OF VIEWPORT IN ROWS
38	(26)	SIGNED	2	PTNWV	WIDTH OF VIEWPORT IN COLUMNS
40	(28)	SIGNED	2	PTNRW	ROW OFFSET FOR WINDOW ON PRESENTATION SPACE
42	(2A)	SIGNED	2	PTNCW	COLUMN OFFSET FOR WINDOW ON PRESENTATION SPACE
44	(2C)	SIGNED	2	PTNRS	ROW SCROLL COUNT FOR VERTICAL SCROLLING
46	(2E)	SIGNED	2	PTNCS	COLUMN SCROLL COUNT FOR HORZ. SCROLLING
48	(30)	SIGNED	2	PTNCCSX	NO. OF HORIZONTAL PELS IN CHAR CELL
50	(32)	SIGNED	2	PTNCCSY	NO. OF VERTICAL PELS IN CHAR CELL
52	(34)	BITSTRING	1	PTNPID	PARTITION ID
56	(38)	BITSTRING	2	PTNFLAG	PARTITION CONTROL FLAGS
		1111 1...		PTNIRM	INBOUND REPLY MODE
		1...		PTNEFM	EXTENDED FIELD REPLY MODE
		.1..		PTNCM	CHARACTER REPLY MODE
		..1.		PTNCMHL	HIGHLIGHT SELECTION BY OPERATOR ALLOWED
		...1		PTNCMCLR	COLOR SELECTION BY OPERATOR ALLOWED
		.... 1...		PTNCMPS	PS SELECTION BY OPERATOR ALLOWED
		1		PTN16BIT	16 BIT BUFFER ADDRESSING MODE SET
		.1..		PTNSCROL	SCROLLABLE PARTITION
		...1		PTNUOM	VIEWPORT UNIT OF MEASURE, CHAR CELL OR PELS
		.... 1...		PTNXATRS	EXTENDED ATTRIBUTE SET
		.... .1..		PTNFLDMD	FIELD MODIFIED

Cross-Reference

	Hex	Hex		Hex	Hex
Name	**Offset**	**Value**	**Name**	**Offset**	**Value**
PTN	0		PTNFLAG	38	
PTNATRCT	4		PTNFLDMD	39	04
PTNATRSZ	14		PTNFVBUF	10	
PTNATRTB	0		PTNIRM	38	80
PTNCCP	16		PTNNXPTN	18	
PTNCCSX	30		PTNPARM	1C	
PTNCCSY	32		PTNPID	34	
PTNCM	38	40	PTNPSBUF	8	
PTNCMCLR	38	10	PTNPSSIZ	6	
PTNCMHL	38	20	PTNRS	2C	
PTNCMPS	38	08	PTNRV	20	
PTNCS	2E		PTNRW	28	
PTNCV	22		PTNSCROL	39	40
PTNCW	2A		PTNUOM	39	10
PTND	1C		PTNW	1E	
PTNDV	24		PTNWV	26	
PTNEABUF	C		PTNXATRS	39	08
PTNEFM	38	80	PTN16BIT	38	01

Response Time Vector Table (RSP)

Size in bytes:	248 (X'F8')
Pointed to by:	Register 6
Function:	The RSP is a common data area used throughout the response time post processor routines. It contains the information specified by the ITPRESP input commands and other information needed for communication between modules.

RESPONSE TIME POINTERS AND PARAMETERS

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	248	RSP	
FLAGS					
233	(E9)	BITSTRING	1	RSPFLAG3	FLAG BYTE 3
		1...		RSPACTUL	ACTUAL RESPONSE TIME
		.1..		RSPTRLST	LIST TRANSACTIONS RECORDS
		1		RSPIGNOR	IGNORE THIS RECORD

Cross-Reference

	Hex	**Hex**
Name	**Offset**	**Value**
RSP	0	
RSPACTUL	E9	80
RSPFLAG3	E9	
RSPIGNOR	E9	01
RSPTRLST	E9	40

Save Area Control Block (SAV)

Size in bytes:	14 (X'E')
Pointed to by:	Chain based on DEVSAVAD
Function:	Save area control block structure.

SAVE AREA CONTROL BLOCK

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	CHARACTER	14	SAVBUFF	ACTUAL SAVE AREA
0	(0)	ADDRESS	4	SAVNXSAV	NEXT SAVE AREA
4	(4)	ADDRESS	4	SAVPVSAV	PREVIOUS SAVE AREA
8	(8)	UNSIGNED	1	SAVNUM	SAVE AREA NUMBER
9	(9)	BITSTRING	1	SAVFLAGS	SAVE AREA FLAGS
		1...		SAVGMSTG	SAVE AREA IN GETMAIN STORAGE
10	(A)	SIGNED	2	SAVBUFSZ	SAVE AREA SIZE
12	(C)	SIGNED	2	SAVDATLN	LENGTH OF LAST DATA SAVED
14	(E)	CHARACTER	*	SAVDATA	ACTUAL SAVED DATA

Cross-Reference

Name	Hex Offset	Hex Value
SAVBUFF	0	
SAVBUFSZ	A	
SAVDATA	E	
SAVDATLN	C	
SAVFLAGS	9	
SAVGMSTG	9	10
SAVNUM	8	
SAVNXSAV	0	
SAVPVSAV	4	

Terminal Control Block (TRM)

Size in bytes:	488 (X'1E8')
Boundary:	Doubleword
Pointed to by:	DEVTRMAD
Function:	The TRM contains information used by WSim in simulating a terminal node.

TERMINAL CONTROL BLOCK

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	488	TRM	TERMINAL CONTROL BLOCK
TRM-DEV FIELDS					
20	(14)	ADDRESS	4	TRMINBUF	ADDR OF INPUT BUFFER
24	(18)	ADDRESS	4	TRMOTBUF	ADDR OF OUTPUT BUFFER
28	(1C)	SIGNED	2	TRMOBUFL	LENGTH OF OUTPUT BUFFER
32	(20)	BITSTRING	1	TRMFLAG1	FLAG FIELDS
		.... .1..		TRMWAIT	WAIT BIT FROM LOGICAL COMPARE
		1		TRMEWAIT	WAITING ON EVENT
MESSAGE GENERATION FIELDS					
36	(24)	SIGNED	2	TRMCURSR	CURRENT CURSOR POSITION
52	(34)	CHARACTER	8	TRMNAME	CONTROL BLOCK NAME
64	(40)	ADDRESS	4	TRMCTRAD	TRM COUNTER ADDRESS
68	(44)	BITSTRING	4	TRMSWCH	USER SWITCHES
85	(55)	ADDRESS	1	TRMUSER	USER DATA BYTE FOR LOGGING
TIMER QUEUE ELEMENT					
228	(E4)	ADDRESS	4	TRMNCBAD	NETWORK CONTROL BLOCK ADDRESS
252	(FC)	ADDRESS	4	TRMLINAD	LINE CONTROL BLOCK ADDRESS
256	(100)	ADDRESS	4	TRMDEVAD	ADDRESS OF FIRST DEVICE
INITIATOR FIELDS					
377	(179)	ADDRESS	1	TRMTYPE	TERMINAL TYPE
		1...		TRMDEV	TYPE IS FOR A DEVICE
		.1..		TRMSNA	SDLC TERMINAL
378	(17A)	SIGNED	2	TRMIBUFL	BUFFER SIZE
414	(19E)	SIGNED	2	TRMUSRLN	LENGTH OF USER DATA
428	(1AC)	ADDRESS	4	TRMUSRAD	POINTER TO USER AREA
432	(1B0)	ADDRESS	4	TRMNXTRM	ADDR OF NEXT TRM
TERMINAL SEQUENCE AND INDEX COUNTERS CONTROL BLOCK					
0	(0)	STRUCTURE	*	TRMCTRS	SEQUENCE AND INDEX COUNTERS
0	(0)	UNSIGNED	4	TRMSEQ	TERM SEQUENCE COUNTER
4	(4)	UNSIGNED	4	TRMSEQCT (*)	TERM INDEX COUNTERS

Constants

Len	Type	Value	Name	Description
TERMINAL TYPES				
1	HEX	30	TRM327T	TCPIP 3270
1	HEX	69	TRMVAPPL	VTAM APPL TRM

Cross-Reference

Name	Hex Offset	Hex Value
TRM	0	
TRMCTRAD	40	
TRMCTRS	0	
TRMCURSR	24	
TRMDEV	179	80
TRMDEVAD	100	
TRMEWAIT	20	01
TRMFLAG1	20	
TRMIBUFL	17A	
TRMINBUF	14	
TRMLINAD	FC	
TRMNAME	34	
TRMNCBAD	E4	
TRMNXTRM	1B0	
TRMOBUFL	1C	
TRMOTBUF	18	
TRMSEQ	0	
TRMSEQCT	4	
TRMSNA	179	40
TRMSWCH	44	
TRMTYPE	179	
TRMUSER	3D	
TRMUSRAD	1AC	
TRMUSRLN	19E	
TRMWAIT	20	04

Glossary, Bibliography, and Index

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 asterisks and 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.

attention identifier (AID). A code that the terminal sends in the inbound data stream to identify the operator action or structured field function that caused the data stream to be sent to the application program. An AID is always sent as the first byte of the inbound data stream. Structured fields in the data stream may also contain an AID.

available. In VTAM*, pertaining to a logical unit that is active, connected, enabled, and not at its session limit.

B

bind. In SNA, a request to activate a session between two logical units (LUs).

C

carriage return (CR). The operation that prepares for the next character to be printed or displayed at the specified first position on the same line. (A)

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 Communications.

CR. Carriage return.

D

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.

ddname. Data definition name.

E

EBCDIC. Extended binary-coded decimal interchange code.

extended attribute buffer (EAB). The buffer in which the extended field attribute for the 3270 kanji display field is stored.

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.

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.

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.

J

JCL. Job control language.

job control language (JCL). A problem-oriented language designed to express statements in a job that are used to identify the job or describe its requirements to an operating system. (A)

L

logical unit (LU). (1) A port through which a user gains access to the services of a network. (2) In SNA, a port through which an end user accesses the SNA network and the functions provided by system services control points (SSCPs). An LU can support at least two sessions—one with an SSCP and one with another LU—and may be capable of supporting many sessions with other logical units.

Loglist Utility. A utility that enables WSim to produce a formatted report of the log data set.

LU. Logical unit.

M

MDT. Modified data tag.

message generation. In WSim, the process of executing WSim statements that generate messages from the resources being simulated by WSim.

message generation statements. The collection of statements that define the actions to be performed by WSim, including message generation and logic testing.

modified data tag (MDT). (1) An indicator, associated with each input or output/input field in a displayed record, that is set ON when data are keyed into the field. The modified data tag is maintained by the display device and can be used by the program using the file. (2) In 3270, a bit in each input field that, when set, causes that field to be transferred to the host system.

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)

Multiple Virtual Storage (MVS). An IBM licensed program whose full name is the Operating System/Virtual Storage (OS/VS) with Multiple Virtual Storage/System Product for System/370*, It is a software operating system controlling the execution of programs.

MVS. Multiple Virtual Storage.

N

NCB. Network control block.

NetView Performance Monitor (NPM). An IBM licensed program that collects, monitors, analyzes, and displays data relevant to the performance of a VTAM telecommunication network. It runs as an online VTAM application program.

network control (NC). In SNA, an RU category used for requests and responses exchanged for such purposes as activating and deactivating explicit and virtual routes and sending load modules to adjacent peripheral nodes.

network control block (NCB). A WSim control block containing information about simulated networks.

network services (NS). In SNA, the services within network addressable units (NAUs) that control network operation through SSCP-SSCP, SSCP-PU, and SSCP-LU sessions.

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.

NS. Network services.

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

PLU. Primary logical unit.

primary logical unit (PLU). In SNA, the logical unit (LU) that contains the primary half-session for a particular LU-LU session. Each session must have a PLU and secondary logical unit (SLU). The PLU is the unit responsible for the bind and is the controlling LU for the session. A particular LU may contain both primary and secondary half-sessions for different active LU-LU sessions. Contrast with secondary logical unit (SLU).

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.

R

record. (1) A set of data treated as a unit (TC97); for example, in stock control, each invoice could constitute one record. (2) In VTAM, the unit of data transmission for record-mode. A record represents whatever amount of data the transmitting node chooses to send. (3) In Series/1*, a portion of a data set accessed at the logical level (GET/PUT).

request/response header (RH). In SNA, control information preceding a request/response unit (RU), that specifies the type of RU (request unit or response unit) and contains control information associated with that RU.

request/response unit (RU). In SNA, a generic term for a request unit or a response unit.

request unit (RU). (1) In SNA, a message unit that contains control information, such as a request code, or function management (FM) headers, end-user data, or both. (2) In DPCX, the smallest unit of data or control information.

resource. (1) Any facility of the computing system or operating system required by a job or task, and including main storage, input/output devices, the processing unit, data sets, and control or processing programs. (2) In the NetView program, any hardware or software that provides function to the network.

Response Time Utility. A utility that enables WSim to analyze response times for activities on the log data set.

response unit (RU). In SNA, a message unit that acknowledges a request unit; it may contain prefix information received in a request unit. If positive, the response unit may contain additional information (such as session parameters in response to BIND session), or if negative, contains sense data defining the exception condition.

return code. A code used to influence the execution of succeeding instructions. (A)

RH. Request header or response header.

RU. Request unit or response unit.

S

script. See WSim script.

SNA. Systems Network Architecture.

STL. Structured Translator Language.

STL Translator. In WSim, a utility that acts as the STL translator and translates STL statements into message generation source statements.

Structured Translator Language (STL). A set of conventions and rules for writing syntactically allowable statements that will create message generation source statements.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through and controlling the configuration and operation of networks.

T

TH. Transmission header.

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. (1) Transmission priority. (2) Transaction program.

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.

transaction program (TP). Any program that uses LU 6.2 communication protocols to communicate with another program. Transaction programs are implemented in WSim using the CPI-C application program interface.

TSO. Time sharing option.

V

Virtual Telecommunications Access Method (VTAM). An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

VTAM. Virtual Telecommunications Access Method.

W

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.

WSim. Workload Simulator.

WSim network. The set of statements defining an entire WSim network, including both the network definition statements and the message generation source statements. Should not be confused with a packet switching network.

WSim script. The set of statements defining an entire WSim network, including both the network definition statements and the message generation source statements.

WTO. Write-to-operator.

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

Index

File Number: S370/4300/30XX-50
Program Number: 5655-I39

Printed in U.S.A.

SC31-8950-00