Entire Net-Work

Application Programming

Version 6.5.2

April 2023

ADABAS & NATURAL

Document ID: WCPMF-APPROG-652-20230324

Table of Contents

Application Programmer's Reference

This document addresses developers of application programs and explains how their programs can communicate interactively with Entire Net-Work. Entire Net-Work provides a user exit interface that allows user-defined programs to be called before connecting or disconnecting a session or before sending or receiving data. Entire Net-Work also provides a Programmable Command Interface that allows application programs to issue Entire Net-Work operator commands and process results from the commands.

Information for the application programmer is provided under the following headings:

User Exit Interface	Describes how to program into the user exit interface.
Programmable Command Interface	Describes how to use the programmable command interface.

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

In addition, you can also access the cloud product documentation via **https://www.softwareag.cloud**. Navigate to the desired product and then, depending on your solution, go to “Developer Center”, “User Center” or “Documentation”.

Product Training

You can find helpful product training material on our Learning Portal at **https://knowledge.softwareag.com**.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at **https://tech-community.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 User Exit Interface

The Entire Net-Work user exit interface allows you to:

- encrypt or compress data before transmission;
- decrypt or decompress data after reception;
- decide to accept or reject a connection from a host that is not predefined to Entire Net-Work; and
- select model link parameters when accepting a connection from a host that is not predefined to Entire Net-Work.

If a user exit is defined, it is called before sending data, after receiving data, before accepting a connect request from a host that has not been defined, and before disconnecting a link to free resources obtained by the exit.

Registers

On entry to the exit, the exit must save the registers in the area provided, which is pointed to by R13. Register contents are listed in the following table:

Register	Contents
R1	Points to a parameter list (described below).
R13	Points to a 72-byte save area.
R14	Contains the return address of the exit.
R15	Contains the entry address of the exit.

Parameter List Format

The following table contains the user exit parameter list format. Also see NETUEXPL DSECT.

Function	Offset	Contents
Function 0		Immediately before sending data:
	+0	Function code = 0
	+4	User context word
	+8	Link name
	+16	Address of message to be sent
	+20	Length of message to be sent
	+24	Address of area to compress/encrypt message
	+28	Length of area to compress/encrypt message

Function	Offset	Contents
Function 4		Immediately after receiving data:
	+0	Function code = 4
	+4	User context word
	+8	Link name
	+16	Address of message received
	+20	Length of message received
	+24	Address of area to decompress/decrypt message
	+28	Length of area to decompress/decrypt message
Function 8		Immediately after accepting a connect request from a host that has not been defined to Entire Net-Work:
	+0	Function code = 8
	+4	User context word
	+8	Generated link name
	+16	Model link name
	+20	VTAM LU name of the remote host (8 bytes)
	+24	TCP/IP Internet host address (4 bytes)
	+28	TCP Port Number of remote host (2 bytes)
	+31	Reserved
	+32	Reserved
Function 12		Immediately before disconnecting a link:
	+0	Function code = 12
	+4	User context word
	+8	Link name

The User Exit

A user exit that compresses/decompresses or encrypts/decrypts must modify the data in the supplied conversion buffer before the data is sent or after the data is received. The user exit must not alter the original message in any manner.

The user exit should be coded to be reentrant although it is never entered more than once during the same execution. With the exception of the area in which the modified message is to be placed, the user exit must provide any storage that it requires.

At offset 4 in the parameter list is a user context word. This word is initially zeroes. The user exit may obtain a work area and place the address of this work area in the user context word. On all subsequent entries to the exit, the user context word remains intact. The user context word on one link is specific to the link and is not related to the user context word on any other link.

When the user exit is ready to accept a connect request for a dynamically added link, the exit may modify the generated link name and the model link name. The link name should be unique within the network due to difficulties in controlling links with operator commands when two or more links have the same name. The model link name must be the name of one of the model links predefined in the Entire Net-Work configuration parameters.

On return from the exit, all registers except R15 must be restored. If the exit modified the length of the message, the message length of the compressed/decompressed message must be updated in the parameter list at offset 28 (X `1C'; field name=NEUXLCNV). Register 15 must contain one of the return codes described in the following table:

Function	Offset	Contents
Function 0		Immediately before sending data:
	+0	Message was not modified (default)
	+4	Message was modified
	+8	Do not send message
	+12	Do not send/discard message and disconnect link
Function 4		Immediately after receiving data:
	+0	Message was not modified (default)
	+4	Message was modified
	+8	Discard message
	+12	Do not send/discard message and disconnect link
Function 8		Immediately before accepting a connect request from a host that has not been defined to Entire Net-Work:
	+0	Accept or reject according to ACCEPTUI setting (default)
	+4	Accept connection regardless of ACCEPTUI setting
	+8	Reject connection regardless of ACCEPTUI setting
Function 12		Immediately before disconnecting a link:
	N/A	Any return code is acceptable.

Restrictions

The following restrictions apply to Entire Net-Work user exits:

- If a user exit is specified on the sending side of a link, an equivalent user exit must be defined on the receiving side of the link. If a link exit is not defined on the receiving side of the link, results are unpredictable and Entire Net-Work may terminate abnormally .
- A user exit on the receiving side of a link that is processing a compressed or encrypted message must do one of the following:

  - Decompress and/or decrypt the message and return with R15 set to 4; or
  - Return with R15 set to 8.

  Returning with R15 set to any other value causes unpredictable results; Entire Net-Work may terminate abnormally.

- The user exit must not alter the original message.

3 Programmable Command Interface

The Entire Net-Work Programmable Command Interface (PCI) is a callable interface that allows application programs to issue Entire Net-Work operator commands and process results from the commands.

- For 3GL applications, the Entire Net-Work subroutine NETPCI is provided. 3GL applications must be linked with both NETPCI and the ADAUSER module.
- For Natural applications, the CALLNAT subroutine USR1070N and the sample program USR1070P are provided in the library SYSEXT.
- For C applications, a sample program is provided.

NODE Statement Parameters

The PCI is controlled using the following NODE statement parameters (for more information, see the section *Entire Net-Work NODE Statement* in the *Entire Net-Work Reference Guide*):

Parameter	Function
ENDCMD	Controls whether operator commands such as NETEND that end Entire Net-Work operation are accepted through the PCI.
LOGSIZE	Specifies the size of a wrap-around buffer used to contain the most recent output written to DDPRINT.
PASSWORD	Enables or disables the Entire Net-Work PCI.
REMCMD	Controls the acceptance of remote PCI calls.

Calling the PCI from a 3GL Program

3GL programs use the NETPCI subroutine to communicate with the PCI. Each call must provide the following ten parameters:

Parameter	Description
TARGET	Target ID of the Entire Net-Work node being addressed.
PASSWORD	Value specified by the PASSWORD parameter on the NODE statement.
FUNCTION	PCI function: COMMAND, CONSOLE, or LOGON.
PLEN	Length of the parameter for the PCI function.
PARM	Parameter for the PCI function.
LINENO	Line number from the CONSOLE function.
RLEN	Length of the PCI reply buffer.
RESULT	PCI reply buffer.

Parameter	Description
RC	PCI response code.
REASON	PCI reason code.

TARGET

Format:	Unsigned binary, length 2.
Input:	The node ID of the Entire Net-Work node to be addressed. If TARGET is set to 0 (zero), the call is presented to the local Entire Net-Work node.
Output:	The node ID of the answering node. This is identical to the input value, except for the LOGON function, or when the input value is 0 (zero).

PASSWORD

Format:	Alphanumeric, length 8
Input:	The password of the Entire Net-Work node addressed. This must match the password specified on the NODE statement in the Entire Net-Work parameter file DDKARTE.

FUNCTION

Format:	Alphanumeric, length 8, left justified
Input:	The function to be executed by the Entire Net-Work node, i.e., LOGON, COMMAND, or CONSOLE

The following functions can be specified and are described in this section:

- LOGON
- COMMAND
- CONSOLE

LOGON

The LOGON function can be used by an application to verify a user-supplied password or determine an Entire Net-Work node ID for a given node name.

- A logon request is accepted by any Entire Net-Work Version 5.7 or later node, regardless of its PASSWORD setting.
- The request is then forwarded to the target node whose node name is specified by PARM.
- After successful logon, the node ID of the responding Entire Net-Work node is returned as the value of TARGET.

The return code (RC) is one of the following:

RC	Meaning
0	LOGON function was successful.
22	LOGON function was rejected. REASON Indicates the cause of the rejection.
148	The node specified by either TARGET or PARM could not be reached. REASON Net-Work node that generated the response code.

COMMAND

The COMMAND function causes the operator command specified by PARM to be executed on the node specified by TARGET.

Output generated during command execution that would normally be written to the local system console is placed in the RESULT output parameter. Each output line is followed by X`15'.

Output generated asynchronous to command execution is not included. For example, in response to a PROBE command, RESULT may contain the following message:

```
NET0136I: PROBE MESSAGE SENT
```

The message is followed by X`15'. However, no indication of the failure or success of the probe message is returned (i.e., no message NET0135.)

The length of the response is returned in RLEN. The return code (RC) is one of the following:

RC	Meaning
0	COMMAND function was successful.
22	COMMAND function was rejected. REASON Indicates the cause of the rejection.
53	Resulting output is longer than the length of RESULT, as indicated in RLEN. The length needed is returned in the parameter REASON.
148	The node specified by TARGET could not be reached. REASON contains the node ID of the Entire Net-Work node that generated the response code.

CONSOLE

The CONSOLE function causes lines from the wrap-around log buffer to be returned in the RESULT parameter.

- Each line is followed by X`15'.
- As many lines are returned as fit into RESULT of length RLEN.
- Output parameter LINENO contains the line number of the first line in RESULT.
- Output parameter RLEN contains the length of the data returned.

The return code (RC) is one of the following:

RC	Meaning
0	CONSOLE function was successful and data was returned.
3	No data was returned; the NODE statement of the target node contains no LOGSIZE keyword.
22	CONSOLE function was rejected. REASON indicates the cause of the rejection.
148	The node specified by TARGET could not be reached. REASON contains the node ID of the Entire Net-Work node that generated the response code.

PLEN

Format:	Binary, length 2
Input:	The length of data in the PARM parameter. The length must not exceed 80 characters.

PARM

Format:	Alphanumeric, length 80 maximum	
Input:	The parameter for the PCI function, as follows:	
	Function	Parameter
	LOGON	The node name of the Entire Net-Work node.
	COMMAND	An Entire Net-Work operator command.
	CONSOLE	None

LINENO

Format:	Binary, length 4
Input:	Reserved for future use. Should be set to 0 (zero).
Output:	For the CONSOLE function only, the line number of the first line received in RESULT.

RLEN

Format:	Binary, length 2
Input:	The maximum length of data that can be received in the output parameter RESULT. The maximum length is 32767. For the LOGON function, RLEN can be set to zero.
Output:	The actual length of the data returned in RESULT.

RESULT

Format:	Alphanumeric, length must be less than 32768
Output:	The response buffer for the COMMAND and CONSOLE functions.

RC

Format:	Binary, length 4
Output:	The return code received from the Programmable Command Interface. For possible values, see the FUNCTION parameter. For any other value returned, see the section *Entire Net-Work Codes* in the *Entire Net-Work Messages and Codes Manual*.

REASON

Format:	Binary, length 4
Output:	For all return codes except 53 and 22, this parameter contains the node ID of the Entire Net-Work node that generated the return code. For return code 53, this parameter contains the length needed to receive the result. For return code 22, this parameter contains one of the following:

Value	Meaning
0	Target Entire Net-Work node is not Version 5.7 or later.
101	Remote commands are not accepted (REMCMD=NO).
103	The PCI is disabled (no value specified for the PASSWORD parameter on the NODE statement).
	104
	Invalid password.
105	Invalid FUNCTION (the value specified is not LOGON, CONSOLE, or COMMAND).

Sample C Program

The following C program calls the NETPCI subroutine.

```
/*    This program is a sample application demonstrating the use      */
/* of function 'netpci' as Programmable Command Interface for         */
/* Entire Entire Net-Work V5.7 or later. This interface allows the           */
/* programmer to 'logon' to a Net-Work node, issue                    */
/* operator commands and receive the reply in a buffer.               */
/* Multiple output lines in the reply buffer are separated by         */
/* logical line-end X'15'. Also, a function is provided to retrieve   */
/* a wrap-around buffer that mirrors recent output to DDPRINT.        */
/*                                                                    */
/*                                                                    */
/* Programming Note:                                                  */
/*    In the data definition below, variable 'target' is initialized  */
/*    to zero.                                                        */
/*                                                                    */
#include <stdio.h>
#define BEGIN (<
#define DO    (<
#define END   >)
#define NE    !=
```

```
main (int argc, char *argv(||))
```

```
BEGIN
```

```
void netpci(unsigned short*, /* Target ID of Entire Net-Work node    */
             char*,           /* Password from NODE statement         */
             char*,           /* Function (COMMAND, CONSOLE, or LOGON)*/
             short*,          /* Length of function parameter         */
             char*,           /* Parameter for function               */
             int*,            /* CONSOLE line number                  */
             short*,          /* Length of reply buffer ( <32768 )    */
             char*,           /* Reply buffer                         */
             int*,            /* Response code                        */
             int*);           /* Reason code                          */
```

```
void neterr(int*,int*);
```

```
unsigned short target = 0;    /* Zero denotes the local Entire Net-Work node */
                              /* See programming note above.          */
char password (|9|)   = "FIVEFOUR";
char function (|9|)   = "LOGON   ";
short int  plen       = 8;
char parm (|81|)      = "DAEFNODE";
long int   lineno     = 0;
```

```
short int  rlen      = 0;
char result (|2001|);
long int  rc, reason;
```

```

char *p, *q;
```

```

/*-----------------------------------------------------------------*/
/*   Have the local Entire Net-Work forward our logon request to     */
/*   node DAEFNODE, and find out about DAEFNODE's target id.         */
/*-----------------------------------------------------------------*/
netpci(&target,password,function,&plen,parm,&lineno,&rlen,
        result,&rc,&reason);
if (rc NE 0)
   DO                      /* It didn't work, let's find out why */
      neterr(&rc,&reason) ;
      return rc;           /* ...and give up */
   END
```

```

/*-----------------------------------------------------------------*/
/*   Variable 'target' is now set to DAEFNODE's node-ID. Send it a   */
/*   HELP command to see what operator commands are available.       */
/*-----------------------------------------------------------------*/
plen = 4;                  /* length of HELP command (excluding NULL)*/
rlen = 2000;               /* usable length of our reply buffer      */
```

```

netpci(&target,password,"COMMAND ",&plen,"HELP",
        &lineno,&rlen,result,&rc,&reason);
if (rc NE 0)
   DO                      /* It didn't work, let's find out why */
      neterr(&rc,&reason) ;
      return rc;           /* ...and give up */
   END
```

```

/*-----------------------------------------------------------------*/
/*   In character string 'result' we now find the help information,  */
/*   each line followed by 0x15. Note that function 'netpci' does   */
/*   not supply a NULL at the end of the result, so if we wanted     */
/*   to use C string functions against 'result', we must set         */
/*   result(|rlen|) to NULL ourselves.                               */
/*-----------------------------------------------------------------*/
p = result;
for (q=p; p < &result(|rlen|); p=++q)
   DO
      while(*++q NE 0x15);          /* find end of a line */
      *q = NULL;                    /* terminate character string */
      printf("%s\n",p);             /* print a line */
   END
END
```

```
/*-------------------------------------------------------------------*/
/*   The error routine. Print an explanation of response code and    */
/*   reason code.                                                    */
/*-------------------------------------------------------------------*/
void neterr(int* nrc,int* nreason)
BEGIN
char *p;
```

```
switch (*nrc)
BEGIN
   case 22:                         /* command rejected */
      switch (*nreason)
      BEGIN
         case 101:
            p = "Remote commands not accepted";
            break;
         case 102:
            p = "Invalid cypher code in ACB";
            break;
         case 103:
            p = "Programmable Command Interface disabled";
            break;
         case 104:
            p = "Invalid password";
            break;
         case 105:
            p = "Invalid PCI function";
            break;
         case 106:
            p = "Invalid length of operator command";
            break;
         default:
            p = "Not an Entire Net-Work V5.7 or later node";
      END
      printf("Command rejected: %s\n",p);
      break;
   case 53:                         /* reply buffer too short */
      printf("Reply buffer length needed is %d\n",*nreason);
      break;
   default:
      printf("Response code %d, generated on Entire Net-Work node %d\n",
             *nrc,*nreason);
END
return;
END
```