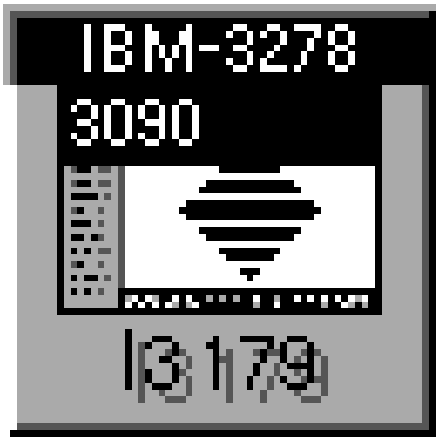
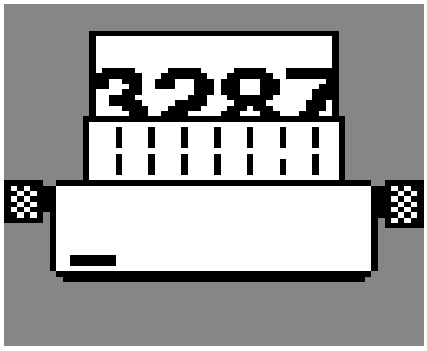
WE-I3179

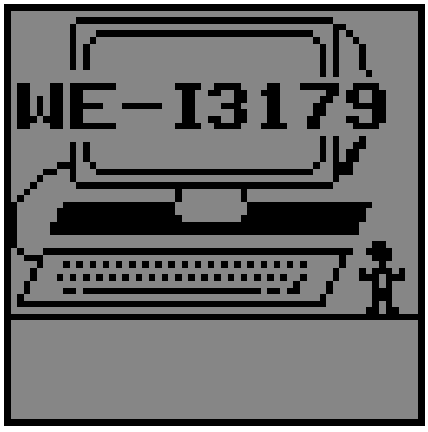
Installation and
User Reference Manual
Version 1.63x - 1.80x Addendum

For NEXTSTEP





and X windows



workstation ag · Rohrstrasse 36/38 · CH-8152 Glattbrugg

The information in this manual is believed to be correct as of the date of publication, however it is subject to change without notice and does not represent a commitment on the part of workstation ag. workstation ag disclaims any warranty of any kind, whether express or implied, as to any matter whatsoever relating to this manual, including without limitation the merchantability or fitness for any direct, indirect, special, incidental or consequential damages arising out of purchase or use of this manual.

The software described in this manual is furnished under a license agreement and may only be installed, used or copied in accordance with the terms of that agreement.

workstation ag · Rohrstrasse 36/38 · CH-8152 Glattbrugg

Table of Contents

1 Preface

1.1 Purpose and audience

This manual is a complement to the Release 1.63x manual. It is intended to all people which need to install, use or maintain the WE-I3179 application. The following informations are contained in this book:

-> New software features since Release 1.63x

-> New or deleted parameters since Release 1.63x

-> Problems corrected since 1.63x

1.2 Summary of content

Chapter 2: "New features and parameters"
presents an overview of the WE-I3179 Release 1.80x new features against Release 1.63x. Also lists new, modified and deleted parameters.

Chapter 3: "New installation procedure"
presents the new installation procedure for WE-I3179.

Chapter 4: "The IND$FILE file transfer utility"
describes how to use this new WE-I3179 feature.

Chapter 5: "The WE-HLLAPI programming interface"
describes the implementation of this API within WE-I3179.

Chapter 6: "The WE-SCRIPT script language"
describes the implementation of this very simple custom Workstation AG script language. Describes how it communicates with a running WE-I3179 and provides some examples how it may be used to make an automatic session logon.

Chapter 7: "X25 connectivity"
describes how to configure and setup the SNA/QLLC/X25 link to the IBM Host.

Chapter 8: "3287 printer emulation"
describes how to configure and use this LUtypes I and III printer emulation which may be used if you have an X25 (SNA) Host connection.

Chapter 9: "Changes for the Keymapper tool"
explains the new features of the KM-I3179 keyboard mapping tool.

Chapter 10: "The WE-LICD licence server program"
explains how to setup and run the floating licence server fro WE-I3179 and how it interacts with WE-I3179.

2 New features, parameters and corrections

2.1 New features

Since Release 1.633, the WE-I3179 product has been expanded with the following important new features whcih will be described in more details in this manual:

-> IND$FILE file transfer has been implemented

-> A programming interface named WE-HLLAPI has been implemented.

-> A script language (WE-SCRIPT) is provided.

-> SNA/QLLC/X25 Host connectivity is supported.

-> For SNA connections, a 3287 printer emulation (both LU tyeps 1 and 3) is provided.

-> The key mapping tool has been enhanced to support more keyboard features.

-> A floating licence server (WE-LICD) is provided with the software.

-> Transparency for text over graphic in the same window has been implemented..

-> Field Mark and Duplicate characters now have the traditionla IBM's look.

-> Better support for SUN type 5 and NeXT new keyboards.

In addition to the previous features, WE-I3179 Release 1.800 now also supports the following new 3270 data stream commands. These commands will not be described in more details here since they are completely explained in IBM's manuals

-> Load Programmed Symbols structured fields are now supported.

-> Graphic escape is now supported.

2.2 New parameters

A brief list of new parameters follows with a brief explaination. A complete description is included in the next chapters

-> **Double_Click_Action** If you set it to 0, a double click will select the word under the cursor.

If you set it to 1, a double click will position the cursor at this position (or at the beginning of the next input field) and transmit the data to the host with an ENTER AID. This is a handy feature for applications where you must often position the cursor and press ENTER to make your selection known to the host.

	NOTE: This setting can be modified with the parameter menu
-> **Cursor_Kind**	If you set it to 0, you will bet a block cursor. If you set it to 1, you will get an underline cursor. NOTE: This setting can be modified with the parameter menu
->**Keyboard_Kind**	You may set values (currently from 1 to 5). See chapter 9 for more informations..
-> **API_Service**	Is used while you intend to start WE-I3179 with an WE-HLLAPI interface. See WE-HLLAPI and WE-SCRIPT chapters 6 & 7.
-> **Host_License_Server**	Is used if you want to get the WE-I3179 licence from the WE-LICD floating licence server. See WE-LICD licence program chapter 10.
-> **License_Server_Policy**	Is used in relation with the Host_License_Server parameter to customize how WE-I3179 and WE-LICD interacts. See WE-LICD licence program chapter 10 for more details.
-> **Xfer_Send_Cmnd**	Is used to store a default (startup value) for the IND$FILE file transfer send command (see IND$FILE chapter for more details)
-> **Xfer_Receive_Cmnd**	Is used to store a default (startup value) for the IND$FILE file transfer receive command (see IND$FILE chapter for more details)

2.3 Modified parameters

The following parameters Have the same meaning as for 1.63x as long as you communicate through TCP/IP. If you communicate through X25, refer to the X25 connectivity chapter 7.

-> Mainframe_Hosts

-> **Mainframe_Service**

-> **Appl_Name**

-> **Term_Name**

-> **Term_Type**

2.4 Deleted parameters

A brief list of deleted parameters follows with a brief explaination.

-> **Point_Mouse**	Deleted without replacement. WE -I3179 now always puts the cursor at the clicked point or at beginning of the next input field

3 New installation procedure

3.1 Extracting the product from the distribution media

The procedure for extracting the products from the distribution media may vary depending on your particular workstation and/or environment. The actual procedure is explained on a sheet attached with your media. Please read this paper and follow the instructions contained therein.

Without going into details, we currently provide 3 different packages kind depending on the machine (operating system) you run:

-> For *NeXTStep based machines*, we provide standard NeXTStep packages on diskettes.

-> For *SUN Solaris 2.x based machines*, we provide standard System V.4 packages.

-> For *all other systems* (SOLARIS 1.x, HP-UX, AIX, ULTRIX), we provide our own packages. These are made with a Workstation AG custom packager.

3.2 What to do when extraction is complete

3.2.1 Reading the README file in the installation directory

This README will provide you with the latest informations about the product and explains how to run the product it the provided Sample configuration file.

3.2.2 Running the product with the Sample configuration file

All products now ship with a Sample configuration file containing a DEMO password allowing the product to run for 15 minutes at each invocation up to some expiration date (see README file). This configuration is setup in such a way that NO Host connection is necessary to startup the product. This allows to make some experiences using WE-I3179 menues and get more confortable with the product. Since the keyboard mapping tool requires No password to run, it is also possible to setup your custom keyboard mapping now and try it with the emulation. Doing so, the product will be much easier to use as soon as the Host connection becomes available.

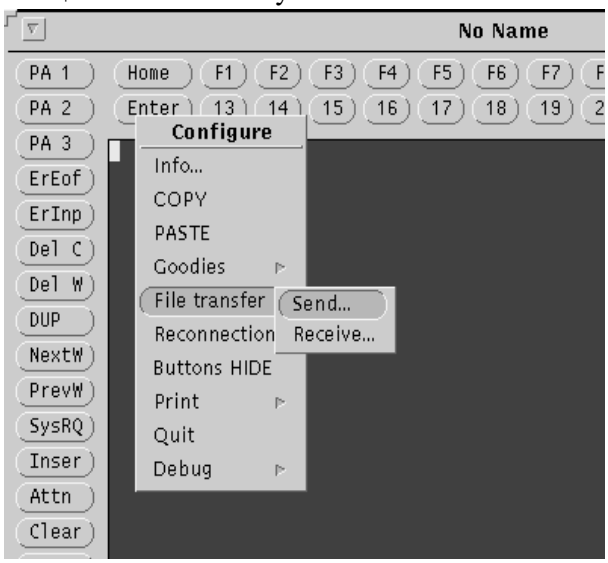
3.2.3 Host connection

Setting up the Host connection for WE-I3179 requires some experience with IBM's systems. Your system administrator will be able to provide you with the necessary parameters depending on your particular host connection. Currently, this may be TCP/IP-Telnet or X25. For setting up these connections, please refer to the according chapters.

4 The IND$FILE file transfer utility

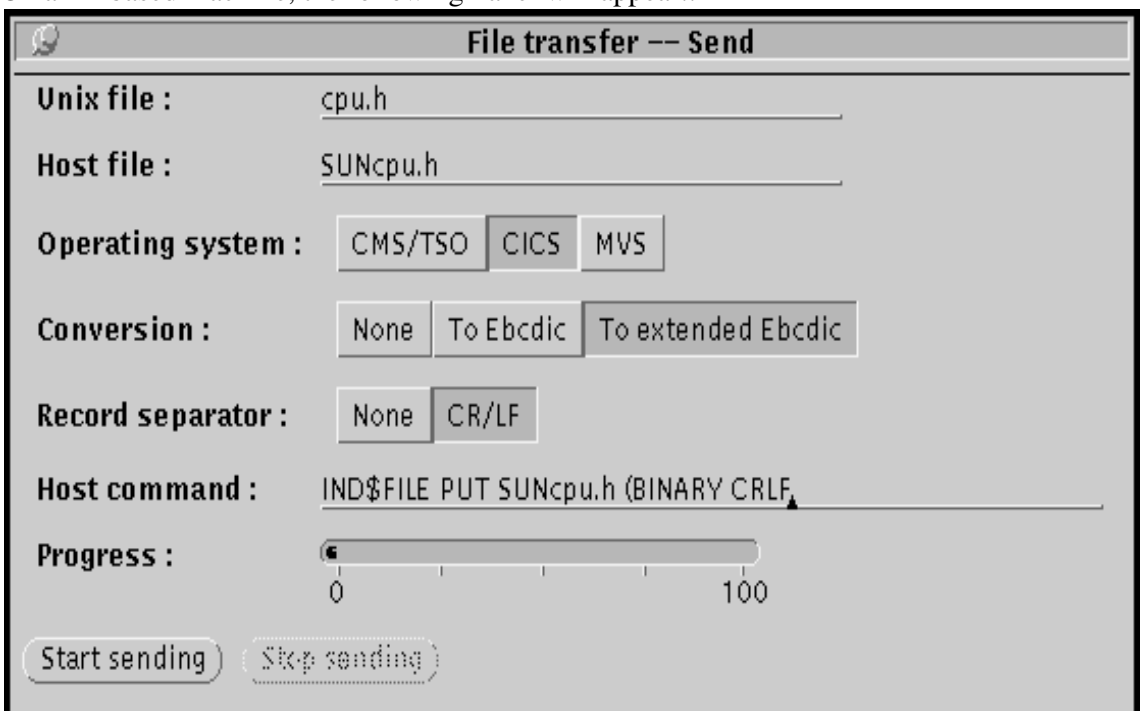
4.1 Starting IND$FILE

IND$FILE is started by the “File transfer” menu as follow:

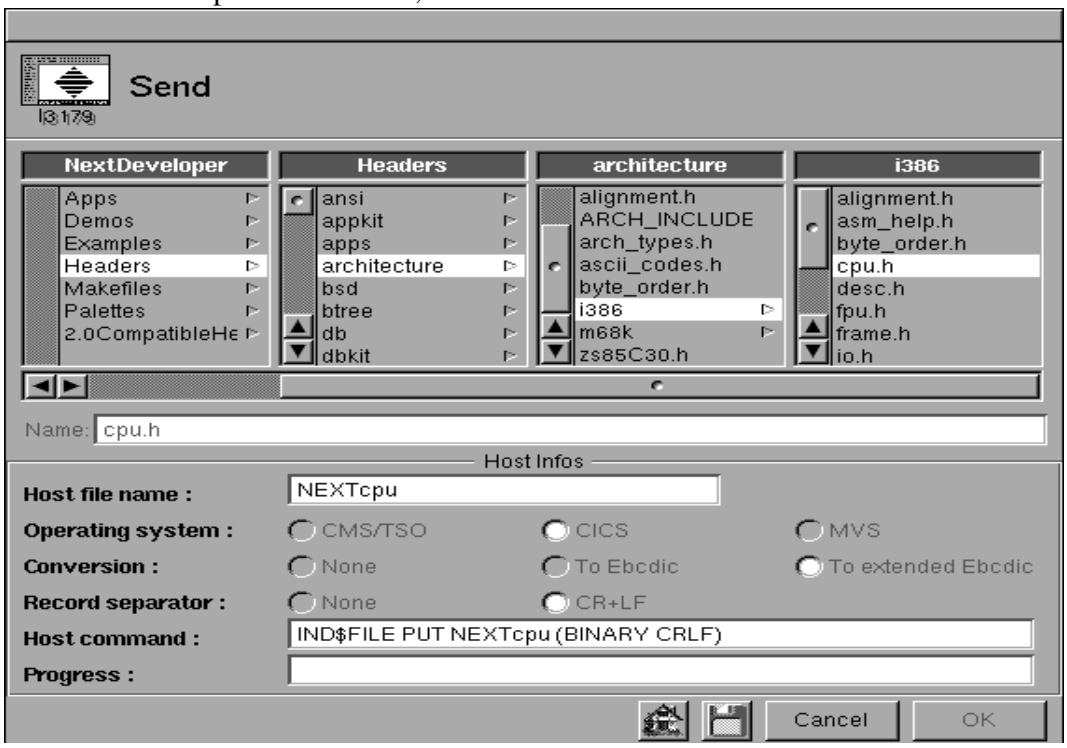


4.1.1 Sending a file with IND$FILE

-> Connect to the Host to which you intend to Send a file and select “Send” in the above menu. On an X based machine, the following Panel will appear:.



While on a NeXTStep based machine, it will look as follow:



There is no functional difference between these 2 panels. NeXTStep simply offers a confortable browser to select the local file. For sending a file, proceed as follow:

(1) Choose the “Unix file” (X) or “Name” (NextStep) to be send.

This is the field where you will enter the local name (and path for X) of the file to send. . It will then appear in the “Host command” field when you hit the tab key.

(2) Choose the “Host file name”.

This is the field where you will enter the name of the file on the IBM Host.

(3) Select an “Operating system”.

You may select one of the 3 proposed options. These options don’t change anything in the behavior of IND$FILE, they merely enable (or disable) conversion and record separator setting fields..

(4) Select “Conversion” and “Record separator”.

You may select one of the proposed options. Depending on the operating system choosen, some or all options may be disabled.

(5) Check if the “Host command” field is correct.

All typing and selections you did before has been converted into a “Host command”. It’s now time to check and modify it if necessary..

(6) Start sending (X) or OK (NeXTStep) fields.

You may click these buttons when you have selected and **carefully checked** (see above) all parameters for the transfer.. This will start the transfer of the file.

(7) Stop sending (X) or Cancel (NeXTStep) fields.

You may click these buttons to abort the transfer..

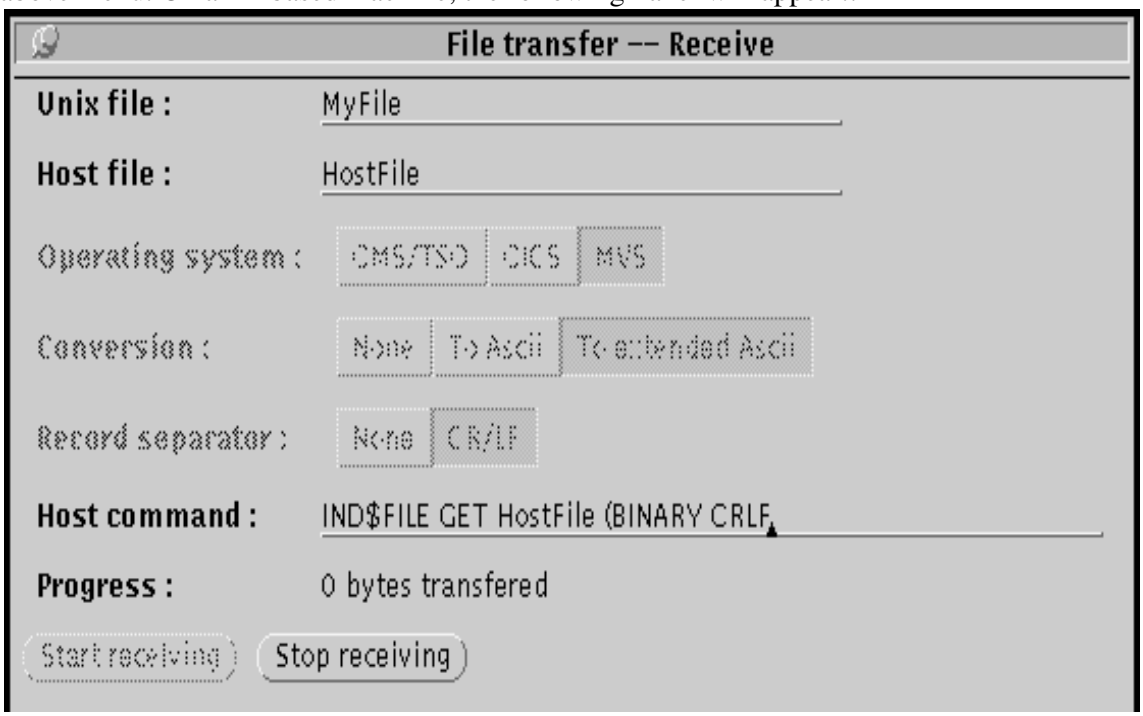
NOTE 1: As you understood, the purpose of all the different fields in these panels is to help you constructing the command which appears in the “Host Command” field. If you need to set other options for the transfer, you may edit this field manually before starting the transfer.

NOTE 2: When you press “Start sending” (or OK), the current field (the field where the cursor stands) of the WE_I3179 window is cleared and the “Host command” field is written into this field just before being transmitted to the IBM Host.

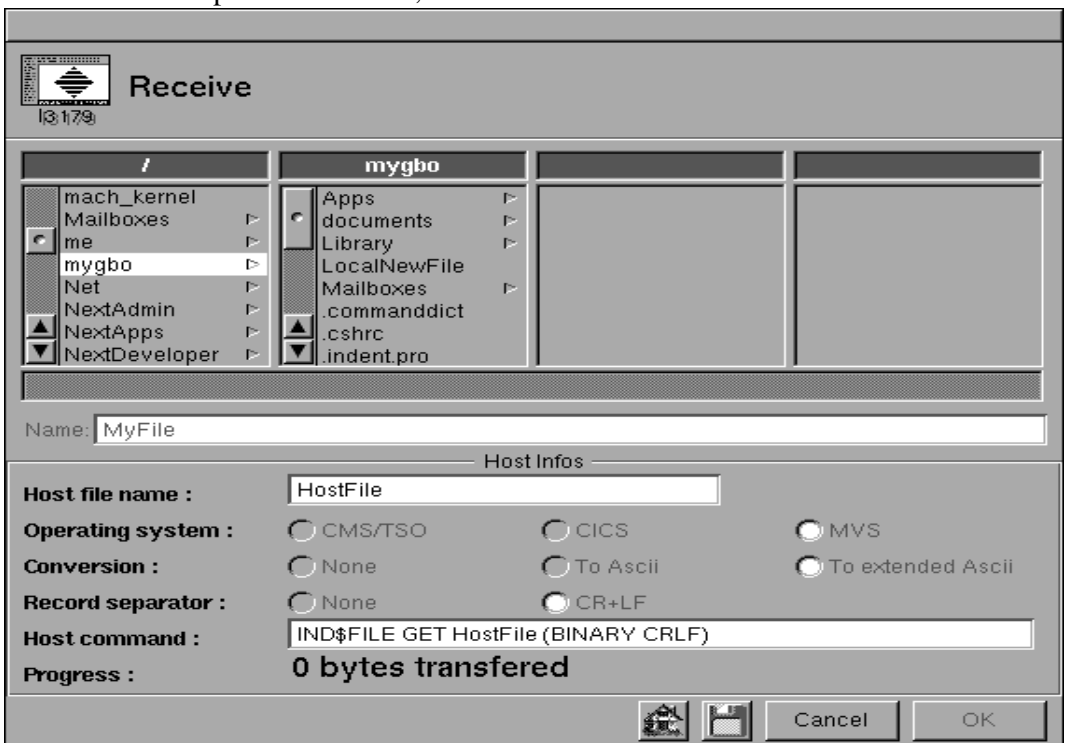
NOTE 3: During the transfer, the progress is displayed in the progress field. Note that you may not return to the main window and type commands while the file transfer is in progress. If you transfer big files and need to work interactively with the host during that time, you may (if your licence allows) start another WE-I3179 emulation

4.1.2 Receiving a file with IND$FILE

-> Connect to the Host from which you intend to Receive a file and select “Receive” in the above menu. On an X based machine, the following Panel will appear:.



While on a NeXTStep based machine, it will look as follow:



There is no functional difference between these 2 panels. NeXTStep simply offers a confortable browser to select the local file. To receive a file, proceed as follow:

(1) Choose the local “Unix file” (X) or “Name” (NextStep) you will give to the received file.

This is the field where you will enter the local name (and path for X) of the file to be received. It will then appear in the “Host command” field. Note that the choosen destination directory must already exist and that you must have rights to create files in that directory.

(2) Choose the “Host file name”.

This is the field where you will enter the name of the file on the IBM Host.

(3) Select an “Operating system”.

You may select one of the 3 proposed options. These options don’t change anything in the behavior of IND$FILE, they merely enable (or disable) conversion and record separator setting fields..

(4) Select “Conversion” and “Record separator”.

You may select one of the proposed options. Dpending on the operating system choosen, some or all options may be disabled.

(5) Check if the “Host command” field is correct.

All typing and selections you did before has been converted into a “Host command”. It’s now time to check and modify it if necessary..

(6) Start receiving (X) or OK (NeXTStep) fields.

You may click these buttons when you have selected all parameters for the transfer.. This will start the transfer of the file.

(7) Stop receiving (X) or Cancel (NeXTStep) fields.

You may click these buttons to abort the transfer..

NOTE 1: As you understood, the purpose of all the different fields in these panels is to help you constructing the command which appears in the “Host Command” field. If you need to set other options for the transfer, you may edit this field manually before starting the transfer.

NOTE 2: When you press “Start receiving” (or OK), the current field (the field where the cursor stands) of the WE_I3179 window is cleared and the “Host command” field is written into this field just before being transmitted to the IBM Host.

NOTE 3: During the transfer, the progress is displayed in the progress field. Note that you may not return to the main window and type commands while the file transfer is in progress. If you transfer big files and need to work interactively with the host during that time, you may (if your licence allows) start another WE-I3179 emulation to do that.

4.1.3 New parameters for IND$FILE

You may use the following two new parameters in you config file(s). These parameters are optional and are intended to give startup values to the above explained fields. They are:

-> **Xfer_Send_Cmnd** MyDefaultHostCommandStringForSendingFiles

Is used to store a startup value for the IND$FILE file transfer send command.

-> **Xfer_Receive_Cmnd** MyDefaultHostCommandStringForReceivingFiles

Is used to store a startup value for the IND$FILE file transfer receive command.

5 The WE-HLLAPI programming interface

5.1 Foreword

-> The WE-HLLAPI programming interface is common to all Workstation AG terminal emulators (currently WE-I3179g, WE-UTSg, WE-D320). Its functions are for the most part similar to IBM's EHLLAPI programming interface

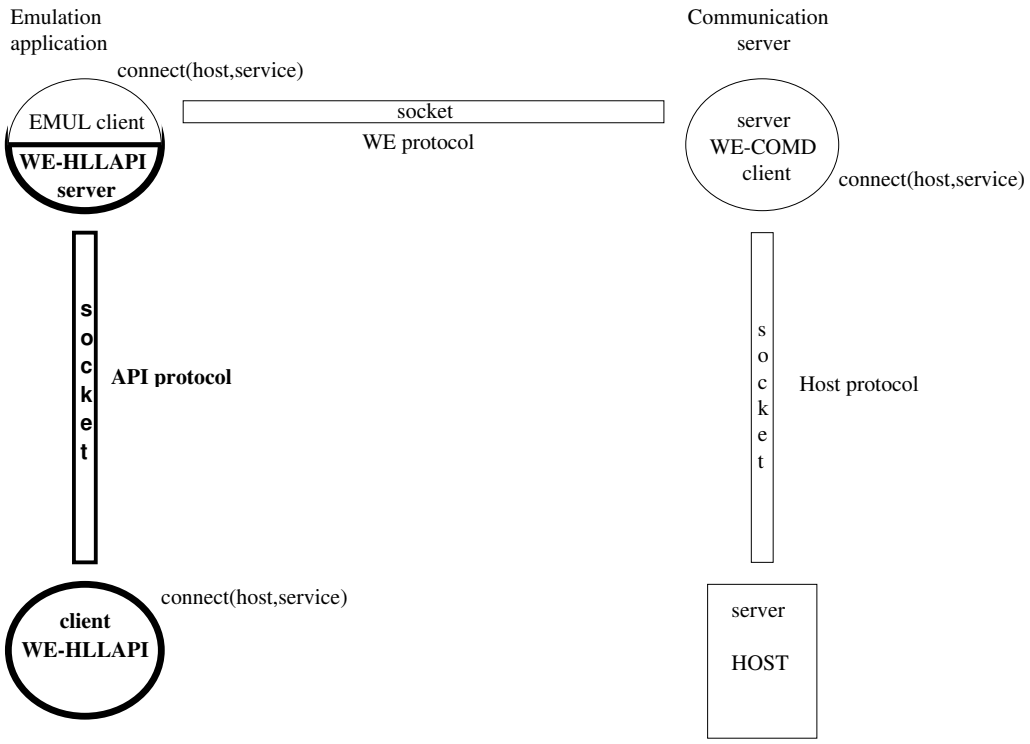
-> WE-HLLAPI is designed to give users and programmers access to the content of the emulator window (sometimes named the host presentation space) with a set of functions that can be called from an application program. Because the emulation handles all network communications to the host system, the user application operates independently of the network protocol.

-> To use WE-HLLAPI, you need an application program written in C language that make calls to WE-HLLAPI library functions.

-> For a detailed description of each "hllc" API call (list in table 2), please refer to IBM's document No SC23-3115-00 (3270 Emulator for the X-Window System) or to any brochure describing the HLLAPI standard API interface.

5.2 Overview

-> The picture below summerizes the client/server architecture of the different WE products. You can recognize that the WE-HLLAPI application is a client against the emulation acting as server.,



5.3 WE-HLLAPI application overview

A WE-HLLAPI runs in 3 phases:

-> Issuing a “XhllapiInit()” call to take contact with the server (The WE-XXX emulation)

-> Making a number of “hllc()” calls to control the WE-XXX emulation behavior

-> Making a ”XhllapiTerm” call to disconnect from the server.

As an example of WE-HLLAPI application, Workstation AG provides customers with the source code of the WE-SCRIPT language (see according chapter). Customers can freely modify and expand this script language. The WE-HLLAPI library, which must be linked to all WE-HLLAPI applications, is available in object format suitable for your particular machine.

5.4 New parameter for the WE-XXX emulation product

-> As you can see in the above picture, a WE-HLLAPI application (the client) needs to “connnect” to a WE-HLLAPI server (the WE-XXX emulation product) before being able to issue any command .This is one of the purposes of the “XhllapiInit()” call.

-> The connection is only possible if the server (the WE-XXX emulation) has registered his service on the network. For this purpose, a new parameter (valid for all WE-XXX emulation products) has been added:

```
API_Service        wagAPI      # Emul will wait for a client connection on this socket
```

-> If you put such a line in one of the config files used by a WE-XXX terminal emulator application, it will initialize itself and then wait for a client to connect to the “wagAPI” service without displaying any window. Of course, you may use any service name of your own instead of wagAPI. Note that you must define this service in the “local” services database (/etc/services) or in some network management system like NIS, Netinfo, ...

-> If all parameters necessary to establish a first host connection were present in the config file, this connection will be attempted but there will still be NO window displayed

5.5 Important remarks

-> When a client (WE-HLLAPI application) connects to the “wagAPI” service, this client takes control over th WE-XXX terminal emulation application. It is the client which decides (through “hllc” library calls) when the WE-XXX emulation should display it’s window and give control to the operator (user of the WE-XXX emulation).

-> The WE-HLLAPI application may decide to display the WE-XXX window, thus allowing the user to interact with the emulator, at any given time. Usually, this is done just before disconnecting form the server through the “XhllapiTerm” library call.

-> A WE-HLLAPI may also decide to give control to the user until a special function is called by the WE-XXX operator. This function is named “M_EXIT” and can be mapped to a button or any key combination of the WE-XXX emulator. When the operator activates this function, WE-HLLAPI (if still connected) regains control over the emulation and can do any operation (ie: autologoff). Control passing between WE-HLLAPI and WE-XXX can occur many times during the excution of the WE-HLLAPI application as long as the WE-HLLAPI applications does not disconnect from the WE-XXX emulator, thus braking the connection for the whole remaining duration of the WE-XXX session.

-> If the user intends to run more than one WE-HLLAPI application at any given time, he must start each couple (WE-HLLAPI application / WE-XXX terminal emulation) with a different “service” name, thus using a different TCP/IP port for each client/server connection..

5.6 library calls, hllc functions, attributes and send key mnemonics

The different library calls, hllc functions, character attributes and send key mnemonics are detailed in the following tables:

.

Table 1: Summary of library calls supported by WE-HLLAPI

Function	Description
XhllapiInit() char *APIServerHost char *APIService unsigned long timeout	Initialize the WE-HLLAPI Name of the host where runs the terminal emulation application. Specify the hllapi service name. Timeout intervall between WE-HLLAPI client and server in ms.
XhllapiTerm()	Disconnect the WE-HLLAPI
XhllapiTermAll()	Terminate the WE-HLLAPI and the we-xxx emulation
hllc() short *fnum char *datastr short *length short *retc	Execute all supported WE-HLLAPI functions WE-HLLAPI function number pointer to the data string length of the data string or parameter return code

Table 2: Summary of hllc functions supported by WE-HLLAPI

Function	Constant Name Description	3270	uts 30	vt300
Change PS Window Name	HA_CHANGE_WINDOW_NAME	Yes	Yes	Yes
Convert Position or Row Col	HA_CONVERT_POS_ROW_COL	Yes	Yes	Yes
Copy Field To String	HA_COPY_FIELD_TO_STR	Yes		
Copy Presentation Space	HA_COPY_PS	Yes		
Copy Presentation Space To String	HA_COPY_PS_TO_STR	Yes		

Table 2: Summary of hllc functions supported by WE-HLLAPI

Function	Constant Name Description	3270	uts 30	vt300
Copy String To Field	HA_COPY_STR_TO_FIELD	Yes		
Copy String To Presentation Space	HA_COPY_STR_TO_PS	Yes		
Find Field Length	HA_DISCONNECT_PS	Yes		
Find Field Position	HA_FIND_FIELD_LEN	Yes		
Query Cursor Loc	HA_QUERY_CURSOR_LOC	Yes		
Query Field Attribute	HA_QUERY_FIELD_ATTR	Yes		
Query Host Update	HA_QUERY_HOST_UPDATE	Yes		
Query Session Status	HA_QUERY_SESSION_STATUS	Yes		
Query System	HA_QUERY_SYSTEM	Yes		
Search Field	HA_SEARCH_FIELD	Yes		
Search Presentation Space	HA_SEARCH_PS	Yes		
Send Key	HA_SEND_KEY	Yes	Yes	Yes
Set Cursor	HA_SET_CURSOR	Yes		
Set Session Parameters	HA_SET_SESSION_PARMS	Yes	Yes	Yes
Start Host Notification	HA_START_HOST_NOTIFY	Yes		
Stop Host Notification	HA_STPO_HOST_NOTIFY	Yes		
Wait	HA_WAIT	Yes	Yes	Yes
Wait Until Key Pressed	HA_WAIT_UNTIL_KEY_PRESSED	Yes	Yes	Yes
Window Status	HA_WINDOW_STATUS	Yes	Yes	Yes

Table 3: Character attributes.

Position	3270 Definition	uts30 Definition	vt300 Definition
0 - 1	00 = Normal 01 = Blink 10 = Reverse video 11 = Underline	00 = Normal 01 = Blink 10 = Reverse video 11 = Underline	00 = Normal 01 = Blink 10 = Reverse video 11 = Underline
2 - 4	000 = Default 001 = Blue 010 = Red 011 = Pink 100 = Green 101 = Turquoise 110 = Yellow 111 = White	000 = Default 001 = Blue 010 = Red 011 = Magenta 100 = Green 101 = Cyan 110 = Yellow 111 = White	000 = Black
5 - 7	Reserved (not used)	Reserved (not used)	Reserved (not used)

Table 4: Send key Mnemonics with Uppercase alphabetic l Characters

ASCII mnemonic	Description	Supported by 3270	Supported by uts30	Supported by vt300
@B	Left Tab	Yes	Yes	**No Action**
@C	Clear display	Yes	Yes	**No Action**
@D	Delete	Yes	Yes	**No Action**
@E	Transmit	Yes	Yes	Yes
@F	Erase EOF	Yes	Yes	**No Action**
@I	Insert	Yes	**No Action**	**No Action**
@L	Cursor Left	Yes	Yes	Yes
@N	New Line	Yes	Yes	Yes
@O	Space	Yes	Yes	Yes
@P	Print screen	Yes	Yes	Yes
@R	Unlock keyboard	Yes	Yes	**No Action**
@T	Right Tab	Yes	Yes	Yes
@U	Cursor Up	Yes	Yes	Yes
@V	Cursor Down	Yes	Yes	Yes
@Y	Caps Lock	**No Action**	**No Action**	**No Action**
@Z	Cursor Right	Yes	Yes	Yes

Table 5: Send key Mnemonics with Lowercase Alphabetic and Numeric Characters

ASCII mnemonic	Description	Supported by 3270	Supported by uts30	Supported by vt300
@0	Cursor Home	Yes	Yes	**No Action**
@1	PF1 / F1 / F1	Yes	Yes	Yes
@2	PF2 / F2 / F2	Yes	Yes	Yes
@3	PF3 / F3 / F3	Yes	Yes	Yes
@4	PF4 / F4 / F4	Yes	Yes	Yes
@5	PF5 / F5 / F5	Yes	Yes	Yes
@6	PF6 / F6 / F6	Yes	Yes	Yes
@7	PF7 / F7 / F7	Yes	Yes	Yes
@8	PF8 / F8 / F8	Yes	Yes	Yes
@9	PF9 / F9 / F9	Yes	Yes	Yes
@a	PF10 / F10 / F10	Yes	Yes	Yes
@b	PF11 / F11 / F11	Yes	Yes	Yes
@c	PF12 / F12 / F12	Yes	Yes	Yes

Table 5: Send key Mnemonics with Lowercase Alphabetic and Numeric Characters

ASCII mnemonic	Description	Supported by 3270	Supported by uts30	Supported by vt300
@d	PF13 / F13 / F13	Yes	Yes	Yes
@e	PF14 / F14 / F14	Yes	Yes	Yes
@f	PF15 / F15 / F15	Yes	Yes	Yes
@g	PF16 / F16 / F16	Yes	Yes	Yes
@h	PF17 / F17 / F17	Yes	Yes	Yes
@i	PF18 / F18 / F18	Yes	Yes	Yes
@j	PF19 / F19 / F19	Yes	Yes	Yes
@k	PF20 / F20 / F20	Yes	Yes	Yes
@l	PF21 / F21 / PF1	Yes	Yes	Yes
@m	PF22 / F22 / PF2	Yes	Yes	Yes
@n	PF23 / F23 / PF3	Yes	**No Action**	Yes
@o	PF24 / F24 / PF4	Yes	**No Action**	Yes
@q	End	**No Action**	**No Action**	**No Action**
@s	Screen Lock	**No Action**	**No Action**	**No Action**
@t	Num Lock	**No Action**	**No Action**	**No Action**
@x	PA1	Yes	**No Action**	**No Action**
@y	PA2	Yes	**No Action**	**No Action**
@z	PA3	Yes	**No Action**	**No Action**

Table 6: Send key Mnemonics For Alphabetic And Symbol Keys

ASCII mnemonic	Description	Supported by 3270	Supported by uts30	Supported by vt300
a through z	Lower case alphbetic characters	Yes	Yes	Yes
Athrough Z	Upper case alphbetic characters	Yes	Yes	Yes
0 through 9	Numeric characters	Yes	Yes	Yes
	Space or Blank Character	Yes	Yes	Yes
~	~	Yes	Yes	Yes
!	!	Yes	Yes	Yes
#	#	Yes	Yes	Yes
$	$	Yes	Yes	Yes
%	%	Yes	Yes	Yes
^	^	Yes	Yes	Yes
&	&	Yes	Yes	Yes

Table 6: Send key Mnemonics For Alphabetic And Symbol Keys

ASCII mnemonic	Description	Supported by 3270	Supported by uts30	Supported by vt300
‘	‘	Yes	Yes	Yes
(	(	Yes	Yes	Yes
)	)	Yes	Yes	Yes
*	*	Yes	Yes	Yes
+	+	Yes	Yes	Yes
’	’	Yes	Yes	Yes
-	-	Yes	Yes	Yes
.	.	Yes	Yes	Yes
/	/	Yes	Yes	Yes
:	:	Yes	Yes	Yes
;	;	Yes	Yes	Yes
<	<	Yes	Yes	Yes
=	=	Yes	Yes	Yes
>	>	Yes	Yes	Yes
?	?	Yes	Yes	Yes
{	{	Yes	Yes	Yes
|	|	Yes	Yes	Yes
}	}	Yes	Yes	Yes
[	[	Yes	Yes	Yes
]	]	Yes	Yes	Yes

Table 7: Send key Mnemonics with @A

ASCII mnemonic	Description	Supported by 3270	Supported by uts30	Supported by vt300
@A@D	Word Delete	Yes	**No Action**	**No Action**
@A@F	Erase Input	Yes	**No Action**	**No Action**
@A@H	System Request	Yes	**No Action**	**No Action**
@A@J	Cursor Select	Yes	**No Action**	**No Action**
@A@Q	Attention	Yes	**No Action**	**No Action**
@A@T	Print Screen	Yes	Yes	Yes
@A@y	Next Word (in a formated PS)	Yes	**No Action**	**No Action**
@A@z	Previous Word (in a formated PS)	Yes	**No Action**	**No Action**

Table 8: Send key Mnemonics with @S

ASCII mnemonic	Description	Supported by 3270	Supported by uts30	Supported by vt300
@S@x	Duplicate	Yes	Yes	**No Action**
@S@y	Field Mark	Yes	**No Action**	**No Action**

Table 9: Send key Mnemonics with Special Characters Keys

ASCII mnemonic	Description	Supported by 3270	Supported by uts30	Supported by vt300
@@	@	Yes	Yes	Yes
@<	Backspace	Yes	Yes	**No Action**

6 The WE-SCRIPT script language

6.1 Foreword

-> WE-SCRIPT is delivered with all Workstation AG emulation products and is aimed to simplify the operators job by allowing to automatisize operations like logging or unlogging to/from a particular system.

-> Since WE-SCRIPT is nothing else than a particular WE-HLLAPI application, all hints and remarks we did in the previous chapter concerning WE-HLLAPI are equally valid for WE-SCRIP. Therefore, we urge you to take a look at the previous chapter before attempting to use WE-SCRIPT.

-> In it's original form (as delivered by Workstation AG), WE-SCRIPT implements a fairly limited set of verbs which are mostly suffisant. As mentioned in the WE-HLLAPI chapter, WE-SCRIPT may be obtained in source code form and is thus expandable by the user. Such enhancements are of course not supported by Workstation AG.

6.2 When should one use WE-SCRIPT or WE-HLLAPI

-> WE-SCRIPT with it's limited functionality (as distributed by WAG) is the best choice for users who can cope with the limited functions set. Writing a WE-SCRIPT script file is a matter of minutes and it can be modified at any time without any recompilation.

-> When WE-SCRIPT limited functionalities are not suffisant, one may choose to write an enterely new WE-HLLAPI application to do the job or to expand the existing WE-SCRIPT (we-hllapi) application to implement new tokens or to expand existing ones. The choice between those 2 approaches will be dictated by the amount of work to do and by the fact that script files will be easier to maintain by a system administrator.

6.3 4. WE-SCRIPT variables.

The SCRIPT application allow using command-line variables.

These variables are defined when running the SCRIPT application. They have the same syntax as in a shell script, that is <$0> for the base name of the SCRIPT, <$1> for the first parameter, <$2> for the second one, etc... .

ie, if you type:

MyScript-s /tmp/ScriptFile -p1 354 -p7 768 -p2 "var" -p3 user

then the variables will have the values:

$0= "ScriptFile"

$1 = 354

$2 = "var"

$3 = "user"

$7 = 768

6.4 Invoking WE-SCRIPT

WE-SCRIPT is invoked as follow:

we-script -s script_file [-p1 par1] [-p2 par2] . [-p32 par32]

where:

-p1...-p32 are command line arguments which are named $1...$32 in the script.

6.5 WE-SCRIPT token list

In it's 1st release, WE-SCRIPT will be delivered with the following tokens:

LogTo	Filename		# Establish *Filename* as log device. Default # is standard error (stderr).
Echo	"String"		# Write *String* to log device
Exit	Number		# End script and return *Number* as exit status
Connect	[host] ServiceNameTime		# Connect to we-xxx hllapi server # through ServiceName. Timeout after *Time*
Disconnect			# Disconnect from we-xxx hllapi server, but # keep the emulation running
Terminate			# Disconnect from we-xxx hllapi server and # terminate the emulation
Call			**# NOT** implemented
Hangup			# **NOT** implemented
Foreground			# Instruct we-xxx to display it's windo**wq**
Background			# Instruct we-xxx to hide it's window
Sleep	Time		# Suspend script execution for *Time* seconds
TestReady	Time		# Wait up to *Time* seconds until kbd unlocks
Match	"String"	[time]	# Search for String in presentation space. # If time present, try each second up to time # and set status accordingly

```
Type        ["String", TOKEN,...]      # Simulate keyboard entries (incl AID keys)
                                       # Example : Type    "@0@V@V@TUserID" will:
                                       # ======  Set the cursor at Home, go two lines down,
                                       #          perform a Tab to go on the first editable
                                       #         field and write "UserID" in it..

If          [ok, bad]                  # Test result of previous operation and
                                       # execute next script line if test matches
Goto        TheLabel                   # Jump to and continue script excution
                                       # at TheLabel

WaitExit                               # Stop script execution here until operator
                                       # call the M_EXIT function by pressing
                                       # the key or the button assigned to it

TheLabel:                              # A token followed by a colon is a label
                                       # You can jump to labels with Goto
#ThecComment                           # A line beginning with a # is a comment line
```

6.6 Using WE-SCRIPT, guidelines

-> Your script should always begin with the “Connect” instruction in order to initialize the communication between your script and the currently running terminal emulation application.

-> While connected, , your script can deal with the emulation application, but by default, the DISPLAY IS NOT VISIBLE . If you want to display the window, you must issue the Foreground instruction . This allow you, (for example in an auto-logon script), to hide all operations until your login is successfully completed. So the window is set to visible only when you are in your account.

-> In case of errors, (ie, a “match” token returns a “bad” result), it is wise to make the window visible with the “Foreground” instruction before displaying some error message within the emulator with the “type” instruction.

-> Making the window visible with the “foreground” instruction also enables the user to type in it. Thus, it is wise to display the window only if the user has to interact with it.

6.7 A commented WE-SCRIPT example

```
# Very simple SCRIPT session
# ====================

# This scripts expects the following command line invocation:
#           we-script -p1 Hostname -p2 ServiceName -p3 ConnectTimeOut

# First, echo all received variables
Echo        "$1, $2, $3"

# Then, try a connection to the emulator
Connect             $1 $2 $3
If                  ok
Goto                TestLogon
Echo                "Sorry, connect FAILED !!!"
Terminate

TestLogon:
Match               "Please Logon"      10
If                  ok
Goto                EnterLogon

NoLogonFromHost:
Foreground
Type                "@0@FMessage from SCRIPT: Sorry, BAD LOGON ! Press EXIT"
Goto                GiveControlToUser

EnterLogon:
Type                "MyCommand@E"
TestReady           30
If                  bad
Goto                NoLogonFromHost

Match               "USERID"  10
```

```
If              bad
Goto            NoLogonFromHost
Type            "MyUserId@T"

GiveControlToUser:
Foreground

# Wait for user to call Exit function (M_EXIT mapped to a key or a button):
WaitExit

# User has pressed exit, perform autologoff:
Type            "Logoff"
Match           "Logoff completed"      10
If              bad
Goto            NoLogoffFromHost
Terminate

NoLogoffFromHost:
Type            "SORRY, automatic logoff failed, do it yourself, please !!!"
Disconnect
```

7 X25 connectivity

7.1 Overview

-> Beside TCP/IP, WE-I3179 may now communicate with the host through a public or private X25 link. For that purpose we have implemented the necessary protocols (SNA/QLLC) in our communication server (WE-COMD).

-> X25 support must be provided by the plattform on which WE-COMD runs. Currently, WE-COMD supports SunLink X25 Release 7 on SUN OS 4.1.x (Solaris 1.0) workstations. Support for other X25 implementations will be provided upon request.

7.2 WE-I3179 modified parameters for X25

No new parameters have been introduced for X25. The same parameters as for TCP/IP are used. Howewer, some of them have another meaning and will therefore be explained thereafter:

Term_Name **dummy**

The “Term_Name” parameter has no meaning while connected through X25. This is still a required parameter, but its value is not used.

Term_Type **2**

The “Term_Type” parameter still contains a number corresponding to a given screen format. Howewer, this is a value which will be used at startup time to create a window with the specified format. Since we use the SNA protocol under X25, we will receive BIND order whcih may change the screen format dynamically. Therefore, the “Term_Type” parameter must be considered as a default screen format value while connected through X25.

Mainframe_Host **MyHostSymbolicName**

The “Mainframe_Host” parameter must contain the symbolic host name corresponding to the X25 NUI (Network User Indentification). This name must correspond to a name found in a line like:

[MyHostSymbolicName, 01712631/123410404001, 50, 2]

in the WE-COMD phone book (see WE-COMD startup chapter below).

Mainframe_Service **dummy**

The “Mainframe_Service” parameter has no meaning while connected through X25. This is still a required parameter, but its value is not used.

7.3 WE-I3179 startup for X25

There are no changes in the WE-I3179 startup procedure for X25. Howewer, read the WE-COMD phone book chapter to learn how LU numbers will be assigned to the connecting WE-I3179 instances.

7.4 WE-I3179 communication status line messages for SNA/QLLC/X25

7.4.1 Configuration related messages

The following messages describes the status of WE-I3179 or WE-I3287 against the WE-COMD performed checks. These statuses may be returned independently of any connection between WE-COMD and the IBM Host..

-> Not connected

Means that WE-I3179 or WE-I3287 is not connected to the communication server (WE-COMD) and therefore also not to the IBM Host. In this state, no data may be sent or received. This is the initial status of WE-I3179 or WE-I3287 before it has established contact with the WE-COMD server.

-> INVALID SNA HOST NAME !

Means that the “Mainframe_Host” parameter found in the config file does not match any host name of the WE-COMD communication server Phone Book (see PhoneBook” file format below for details). You will have to correct this entry in the config file and restart WE-I3179 or WE-I3287.

-> ALL PORTS OF HOST USED !

Means that there is no free LU on the specified “Mainframe_Host” connection to assign to this WE-I3179 or WE-I3287 client. You may either try later (when other clients have disconnected) or use another configuration file with a different “Mainframe_Host” entry (if available).

-> INVALID ACCESS TO PORT !

Means that there is no free LU on the specified “Mainframe_Host” connection with the proper acces rights (see PhoneBook” file format below for details) to assign to this WE-I3179 or WE-I3287 client. You may either try later (when other clients have disconnected) or use another configuration file with a different “Mainframe_Host” entry (if available).

7.4.2 SNA related messages

The following messages describes the status of the SNA connection between WE-COMD and the IBM Host. They are result of the dialog occuring between the two.

The PU (Physical Unit corresponding to a PhoneBook Host entry) related messages concern all LU’s belonging to this PU and will therefore be displayed by all LU’s using this PU (LU’s listed under the same PU definition in the PhoneBook).

The LU (Logical Unit) related messages are specific to a single LU and are therefore displayed only on this concerned LU (WE-I3179 or WE-I3287 instance).

-> PU session established

Means that the PU has been activated by the Host (may only occur after successfull X25 and QLLC link establishement) .

-> PU session broken

Means that the PU has been deactivated by the Host . This should normally not occur except after the last client has disconnected from that PU.

-> LU session established

Means that the LU has been activated by the Host . **This is the normal working status**.

-> LU session unbound

Means that the LU has been deactivated by the Host . This is an intermediate status while changing from an application to another (sna Unbind / Bind sequence).

-> LU session broken

Means that the LU has been deactivated by the Host due to some error.

7.4.3 X25 related messages

The following messages describes the status of the QLLC/X25 connection between WE-COMD and the X25 public or private network. They are result of the dialog occuring between the two and concern all clients using this particular PU (see PhoneBook).

-> Opening X25 connection

Means that WE-COMD is trying to open the link to the Host commuinication controller.

-> CALL RESET ERROR (N1, N2)

Means that the link between WE-COMD and the Host commuinication was broken due to some error. N1 and N2 are the X25 error codes documented in the X25 CCITT documentation.

-> CALL CLEAR ERROR (N1, N2)

Means that WE-COMD attempt to open the link to the Host commuinication controller failed. N1 and N2 are the X25 error codes documented in the X25 CCITT documentation.

-> ERROR, NO HOST connection !

Means that WE-COMD attempt to take contact with the X25 service on the workstation has failed. Please, check your X25 service installation.

7.5 WE-COMD startup for X25

WE-COMD heen enhanced to support SNA/QLLC/X25 connectivity. To startup WE-COMD in SNA/QLLC/X25 mode, you must issue the following command:

we-comd ServiceName [-sm] [-lc] [-lp LogFilePath] -t sna-x25 -pb MyPhoneBook

The -sm, -lc and -lp optional parameters are fully explained in the 1.63x manual. The “sna-x25” new type for the -t option will instruct WE-COMD to start up in SNA/QLLC/X25 mode. The option -pb (pb stands for PhoneBook) is new and mandatory for X25. Of course, you may use any file name after the -pb option instead of “MyPhoneBook”.

7.6 WE-COMD “PhoneBook” file format

The PhoneBook file is an ASCII file (you may create it with any program editor) which contains tokens used by WE-COMD while running in SNA/QLLC/X25 mode. It’s purpose are:

-> Defining symbolic host name corresponding to X25 NUI addresses / XID’s

-> Defining the LU types

-> Defining how LU numbers will be assigned to clients (WE-I3179 or WE-I3287)

Thereafter, we will list a sample PhoneBook file and explains the tokens using the listed entries:

This is a sample for the SNA phone book used by theWE-COMD server. It is used

only while using WE-COMD in X25 mode.

[HostSymbolicName_1, 01733333/123410404320, 32, 2] # Definition for a first PU

< 2, N, nobody >	This LU will never be assigned
< 3, N, nobody >	This LU will never be assigned
< 4, C, host:jupiter >	Will accept WE-3287 client running on host “jupiter” only. Will accept to run as LU type 1 or 3 (depending on BIND received).
< 5, 1, host:saturne >	Will accept WE-3287 client running on host “saturne” only. Will accept to run as LU type 1 exclusively.
< 6, 3, host:venus >	Will accept WE-3287 client running on host “venus” only. Will accept to run as LU type 3 exclusively.

< 7, 2, user:suzan+peter >	Will accept WE-I3179 client run by user "suzan" or "peter" only
< 8, 2, user:steve >	Will accept WE-I3179 client run by user "steve" only
< 29, 2, appl:WordProc >	Will accept WE-I3179 client whose "Appl_Name" parameter has been set to "WordProc" only

[HostSymbolicName_2, 01755555/123410404321, 16, C] # Definition for a second PU

 # end of the definitions

When we look at that file, we can deduct the following rules:

Rule 1: general syntax:

-> A PU definition is included in [ ] and comes just before the LU definitions belonging to him.

-> A LU definitions is included in < >

-> LU definitions belongs to the just preceeding PU definition.

-> Comments may appear anywhere except in front of or within a PU or a LU definition.

-> There are NO comment introducer character

Rule 2: PU definition syntax:

-> [IBMHostSymbolicName, BlockIdXid/X25NuiNumber, NumberOfLu, DefaultLuType]

Where:

-> A comma must stand between each field described below. All fields are mandatory.

-> The IBM symbolic name is arbitrary and is matched to the "Mainframe_Host" parameter in the config file of connecting WE-I3179 or WE-I3287 client.

-> The BlockID, XID and X25NuiNumber are IBM and X25 provided numbers. There may be NO spaces or tabs in this field. BlockID must come first, followed by XID, a / and finally the X25 NUI. This syntax is mandatory and you must stricly adhere to it.

-> The number of LUs field describes how many LU's are addressable on that particular PU. LU 0 and 1 are comprised in that number but are reserved for the internal use of WE-COMD.

-> The default LU type is the LU type used for all LU's on that PU. We recommend that you set it to the type corresponding to the most common type of LU on the PU. You can override this default type with a LU definition (see below).

Rule3: LU definition syntax:

-> < LuNumber, LuType, WhoOrWhereOrWhat >

Where:

-> A comma must stand between each field described below. All fields are mandatory.

-> *LuNumber* is the number of the LU we want to define. This number may be anything between 2 and (NumberOfLu - 1) defined for the corresponding PU.
Example: If NumberOfLu is 32, you may specify LU numbers from 3 to 31.

-> *LuType* may be one of the following:

N for not connectable (not usable LU's)

1	for LU type 1 (WE-I3287 SCS printers)
2	for LU type 2 (WE-I3179 terminal)
3	for LU type 3 (WE-I3287 Data Stream Compatibility printer)
C	for LU types 1 or 3 (WE-I3287 SCS or DCS printer)

-> *WhoOrWhereOrWhat* may be one of the following:

nobody	=> NO client should connect to this LU
anybody	=> ANY client can use this LU (if free)
user:Name.1+Name.2+Name.n	=> ONLY clients whose user's name matches one of the => name in the list can connect to this LU. This is useful to => reserve a LU(mainly terminals) for well defined users.
host:Name.1+Name.2+Name.n	=> ONLY clients whose machine name matches one of the => name in the list can connect to this LU. This is useful to => reserve a LU(mainly printers) for well defined Unix hosts
appl:Name.1+Name.2+Name.n	=> ONLY clients whose application name matches one of the => name in the list can connect to this LU. This is useful to => reserve a LU(mainly terminals) for well defined config => files where Name.x appears beside the "Appl_Name" => token.

NOTE: The above user, host and appl tokens may not be combined and can't contain spaces

8 WE-I3287 printer emulation

8.1 Overview

-> The 3287 printer is an IBM printer supporting both SCS (Sna Character Stream) and DCS (3270 Data Stream Compatibility) protocols. Both protocols are implemented in the WE-I3287 product.

-> Against WE-COMD (the communication server), WE-I3287 is a client like WE-I3179. The printer emulation is a separate product which may run in parallel with terminal emulations over the same WE-COMD instance.

-> Because IBM currently only support LU types 1 and 3 over SNA protocols, the WE-I3287 product can only be used while the WE-COMD communication server communicates with the host through any SNA protocol stack (cureently only QLLC/X25).

8.2 Licence for WE-I3287

-> The licence for WE-I3287 is the same as for WE-I3179. This means that the licence (password) you get for WE-I3179 is equally valid for WE-I3287.

-> If you use the floating licence server (WE-LICD), WE-I3287 register itself as a character only WE-I3179 emulation. Therefore, it permanently uses a licence while running.

8.3 Starting WE-I3287

-> The startup procedure for WE-I3287 is exactly the same as for WE-I3179. This means that the possible command line parameters are common and that you may apply the WE-I3179 starting procedure without any restrictions.

-> A simple invocation of WE-I3287 could be:

we-i3287 SampleX11.I3287_config

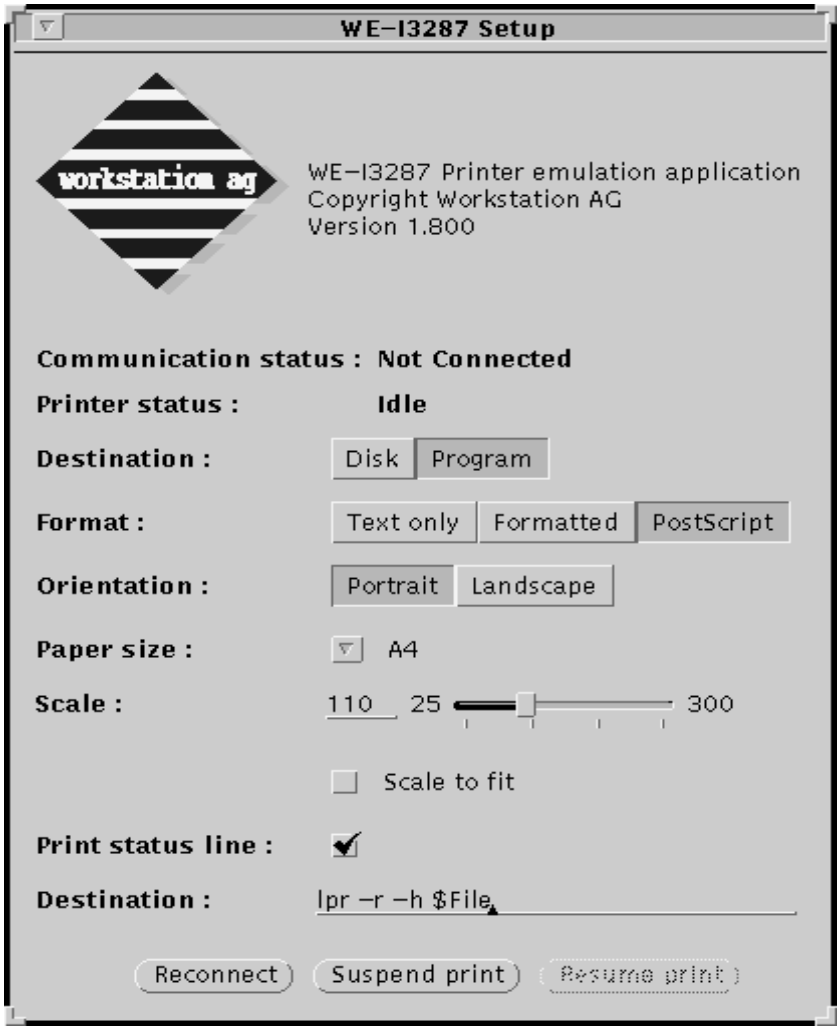
Where: “SampleX11.I3287_config” is the name of the configuration file to use.

-> As for WE-I3179, you are not limited to a single configuration file for WE-I3287. There may be as many as 4 configuration files read one after each other. For details about this and how it can be used, please refer to the WE-I3179 starting procedure.

-> While started, WE-I3287 displays its setup window on the screen (see below). With this setup window, you may change many print parameters which may be set by default or by values read from the configuration file(s).

-> Most of the entries in the setup window are exactly the same as the correspoonding entries of the “Print Format” menu of WE-I3179. They will therefore not be discussed in details here.

:



8.4 WE-I3287 setup panel options

(1) Communication status

The current communication status is displayed in that field. The messages are the same as for the status line communication field of the WE-I3179 emulation in SNA mode. Please refer to the X25 connectivity chapter 7 for details.

(2) Printer status

The current WE-I3287 printer status is displayed in that field. This may be:

-> **Idle**

Means that WE-I3287 is currently ready to accept a new print request from the Host..

-> **Receiving**

Means that WE-I3287 is currently receiving print data from the Host.

-> **Printing**

Means that WE-I3287 is currently printing the data received from the Host. This may or may not produce paper output depending on the Destination choosen for printing.

(3) <Reconnect button>

At startup, WE-I3287 (like WE-I3179) attempts a connection to the Host (through WE-COMD). If, for any reason, the communication breaks (see communication status field), a reconnection may be attempted by pressing this button.

(4) <Suspend print> and <Resume print> buttons

These buttons may be used to stop or restart the data flow coming from the host without breaking the communication. While suspended, the printer LU remains active but busy against the host.

8.5 WE-I3287 configuration file format

The figure below displays a typical WE-I3287 configuration file:

```
# WE-I3287 DEFAULT PAGE SIZE PARAMETERS
Screen_Cols       80            # Page width in characters
Screen_Rows       66            # Page height in lines
#
# WE-I3287 MISCELLANEOUS OPTION
Appl_Name         dummy         # Dummy parameter for compatibility
Term_Name         dummy         # Dummy parameter for compatibility
Term_Type         dummy         # Dummy parameter for compatibility
#
National_Cset     INTERNATIONAL # Code page 500 will be used
Lock_Setup        0             # Don't lock this config file while used
Invert_Bold       0             # Don't invert bold and normal characters
#
# WE-I3287 SERVER AND HOST CONNECTION OPTIONS
Comd_Host         localhost         # The machine on which <we-comd> runs
Comd_Service      wag3270           # Service name of <we-comd> server
Mainframe_Host    MySymbolicName    # The symbolic name of the IBM machine
                                    # known by <we-comd> fromPhoneBook
Mainframe_Service dummy             # Dummy parameter for compatibility
#
# WE-I3287 PRINT PARAMETERS
Disk_Print        ~/we-i3287.$ext   # Destination for Disk print
Print_Program     lpr -r -h $File   # Destination for Program print
Print_Dest        Disk              # Destination is DISK
Print_Format      Postscript        # Format is Postscript
#
# The following useful only if "Print_Format" IS "Postscript"
#
Print_Scale           0             # 0 means scale to fit
Print_Output_Size     A4            # Paper size
Print_Generic_File    ps_generic.dat # PostScript header + trailer file
#
# WE-I3287 PASSWORD
#
Pass_Word:@7?B?E=DCCMDP8@DA9=:C7A9K5O6KB<)

# Host_License_Serverlocalhost    # For customers with WE-LICD license service
```

As you can see, the parameters are the same as for WE-I3179 and will threfore not be explained again here. The only exceptions concern the two following parameters:

-> Screen_Cols

Has actually little to do with “screen”. This parameter contains the default page (paper) width in characters. Since the Sna BIND order may specify other values, this is only a default.

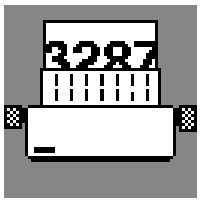
-> Screen_Rows

Has actualy little to do with “screen”. This parameter contains the default page (paper) height in lines. Since the Sna BIND order may specify other values, this is only a default.

NOTE: *Some or all of the required DUMMY parameters may disappear in a further Release.*

8.6 WE-I3287 Icon

The WE-I3287 look s as follow:



9 Changes to the Keymapper tool

9.1 Overview

Some design changes have been made to the Keymapper tool for Release 1.800 and concerns all emulators (WE-UTS, WE-D320 and WE-I3179). When both the emulations and the corresponding tool are concerned, they will be treated concurrently in this chapter.

9.2 The new <Keyboard_Kind> emulator option.

A new option has been introduced for all emulators. For now, only SUN and NeXT computers are concerned. It's syntax is as follow:

Keyboard_Kind **N**

Where N can be:

1 (default)	SUN type 4 keyboard NeXT 1st keyboard (all black)
2	NeXT 2nd keyboard (green power key)
3	Standard MAC keyboard on NeXT
4	Extended MAX keyboard (F keys) on NeXT
5	SUN type 5 keyboards

For the emulation, it is suffisant to put the correct option in the configuration file used. The key mapping tool must be started with the new <-kk N> option.

EXAMPLE:

km-i3179 NeXT-Us.I3179_keymap -kk 4

This line would allow one to setup a good keyboard mapping for the WE-I3179 emulator on a NeXT computer equipped with a Macintosh extended keyboard. Of course, this user needs to put the following in his config(s) file(s) to effectively use this mapping:

Keyboard_Kind **4**

9.3 Keyboard Layout show mode .

We have changed the keyboard layout show mode (goodies menu of the meulators) behavior as follow:

-> If a comment has been entered in the comment field with the key mapping tool, it is shown.

-> If no comment has been entered for that function, the key codes are shown.

Previously, both were shown. This was not always good, because the key codes may be very confusing for the normal user. So, we recommend to system administrators to put comments for each key combination which is not self explanatory (or even cryptic...)

9.4 New functions mappable to the emulator keyboard or buttons.

The following mappable functions have been added to WE-I3179 and can be found in the list displayed by the KM-I3179 key mapping utility. The same names may also be used to map functions to the buttons around the window.:

-> M_PRINT	Will make a hardcopy using the current format settings
-> M_INS_BACKSPACE	Works like the standard (IBM) Backspace function while insert mode is active (Go back 1 position and delete character at that position). The standard IBM backspace function is of course still available under the M_BACKSPACE name.

9.5 Miscellaneous concerning key mapping tool and emulator.

Thereafter, a brief list of the enhancements to the keyboard mapping since Release 1.63x.

-> When you use keyboard "ESCAPE sequences" (see keyboard mapper), you now get a "compose" indication on the emulator status line. This is a useful information for the operator.

-> An ESCAPE sequence may now be aborted by pressing ESCAPE again.

-> On NeXTStep, the definition file for the keyboard layout (“keylay.dat” file) is always loaded at application startup if it is in the same folder as the application. This helps starting the keyboard mapping tool with a double click on the key mapping file (with the corresponding .D320_keymap, .I3179_keymap, .UTS_keymap extension).

-> The NeXT Keypad keys generate a new “Num-Lock” modifier. Typing the '2' key on the main part of the keyboard will generate "2", typing '2' on the numeric keypad will generate "Num-Lock 2".

-> All X modifiers are handled. They're displayed as 'Meta[x]' where x is a digit from 0 to 9 (actually only 0 to 2) in the keymapper application.

-> Some "special" keys (keys that have a corresponding X name) that had been omitted are now available (/Prior, /Next, ...).

Changes to the Keymapper tool

-> A new value for the "Keyboard_Kind" parameter has been created to support the SUN type 5 keyboards. The problem was to distinguish between cursor keys and cursor functions on the keypad with the num-lock modifier. If you set the "Keyboard_Kind" to 5, the keypad cursor keys will return new codes (of format '$...'). The same option is available for the key-mapper tool.

-> The compose key (SUN keyboards) now works. This is useful for people whih have US keyboards and must type european characters. It is NOT necessary to use the key mapping tool for this purpose since such “composite” characters are already mapped by default in the terminal emulators.

10 The WE-LICD licence server program

10.1 Overview

WE-LICD is a Workstation AG custom floating licence server delivered free of charge with all software products and allowing flexible licencing over your network. Like all such products, WE-LICD should be run on a secure machine running at all time and attainable by all client machines. The installation of WE-LICD is very easy. All what it needs is a simple password file containing a single entry, the password provided by Workstation AG.

10.2 Purpose of the password

The password (an ASCII string containing 34 characters) contains the following informations:

-> The product kind (WE-UTSc, WE-UTSg, WE-D320, WE-I3179c, WE-I3179g, W-PLAN,...)

-> The number of instances of the product which may run at any given time.

-> The licence run time (for DEMO products)

-> The licence expiration date

10.3 The password file

As previously explained, the password file contains a single entry, the password. It may also contain some comments of your own.

```
# A WE-LICD "config" file MUST contain the following entry:
# ========================================

Pass_Word               PutYourLicenceServerePasswordHere
```

The format of the single “Pass_Word” parameter and from the comment lines is the same as for all emulator products. Please refer to that documentation for details.

10.4 New parameter for the emulation products using WE-LICD

To tell the emulation that a Network Licence Server (WE-LICD) must be used, the following entry must be added into your config file(s):

Host_License_Server HostName

Where: “Hostname" is the name of the host where the WE-LICD server is running.

Important Remark: If a line like:

Pass_Word G;9C9:9K8=CBYOIR-COUCOU7=4=7>:LA

is also present in the config file(s), the "Host_License_Server" entry takes precedence over it. Note that this may change in a further release. Therefore, we discourage you to have both a "Pass_Word" and an "Host_License_Server" entry in your config file(s). The best in to comment out (with a # at the first line position) the unused entry.

10.5 Running WE-LICD

The licence server is started as follow:

we-licd ConfigFileName [-lc] [-lp LogFilePath]

Where:

ConfigFileName	must be the first parameter. This is the name of the file where the Network Licence Password is stored.
-lc	is optional (means log to console). Normally, WE-LICD creates a file (see -lp option below for possible alternate path) to store it’s logging information. This file is stored in </tmp/we-licd.PID> where PID is the process ID of the WE-LICD instance. REMARK: Depending on your particular system, writing messages on the console may destroy the appearance of your graphical environment. Mostly, this can be repaired with the <refresh> function of your window manager.
-lp LogFilePath	may be used to specify a custom pathname for storing the <we-licd.-PID> logging file.

REMARKS:

->The WE-LICD floating licence server serves only one product type at a time. You may start more than one instance of WE-LICD on the same machine, each for a different product.

-> RPC's are used between client (WE-UTS, WE-D320,...) and WE-LICD. The client product automatically "searches" the server for a WE-LICD instance on the server machine that matches it's type.

-> When a client terminates normally (not killed...), his slot (licence) is immediately freed in the WE-LICD server. Otherwise, the server will automatically free it after about 5 minutes.

10.6 New emulator parameter for customizing the use of WE-LICD

There may be users which have licences for 2 levels of the same product. Let say for WE-I3179c (character terminal emulation) and for WE-I3179g (graphic terminal emulation). If such users have floating licences for both product, they will have to run 2 instances of WE-LICD (one for each licence). Howewer, the need to decide which user will get which kind of licence will probably arise. For this, they may put the following entry in their config files:

License_Server_Policy Policy

Where: “Policy" may have 3 different values as follow:

Best	The emulation will try to get a licence from the WE-LICD serving the best possible licence (in our case graphic). If no such licence is available, it will try the less valuable one before failing if none is available. This is the default behavior.
Graphics	The emulation will try to get a licence from the WE-LICD serving the a graphic licence only. If none is available, it will fail.
Text	The emulation will try to get a licence from the WE-LICD serving the a text licence only. If none is available, it will fail.

Example:

-> 1st WE-LICD instance serving 4 WE-UTSg (graphic) licenses

-> 2nd WE-LICD instance serving 10 WE-UTSc (text) licences

The behaviour will be as follow:

For users with License_Server_Policy set to Best

-> The clients connecting first will get the 4 graphic licences and clients coming later will get the text only licences.

For users with License_Server_Policy set to Graphics

-> These clients will either get a graphic license or no licence at all if no more are available.

For users with License_Server_Policy set to Text

-> These clients will either get a text license or no licence at all.

REMARKS:

-> By setting this parameter to one of these 3 values depending on the client, you will be able to decide in advance who will qualify for either kind of licence.

-> The “License_Server_Policy” parameter is only useful if both WE-COMD instances run on the same machine because there may be only one “Host_License_Server” parameter in a configuration file.