WE-D320

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

For NEXTSTEP

and X windows

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

Copyright© 1992, 1993 by workstation ag. All rights reserved.

Reproduction of this document, in part or whole, by any means, electronic, facsimile or otherwise, is prohibited, except by written permission from workstation ag.

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.

NeXT, the NeXT logo, NEXTSTEP and Workspace Manager are trademarks of NeXT, Inc. UNIX is a registered trademark of UNIX Systems Labs. DEC, VT100, VT220, VT320 are registered trademarks of Digital Equipment Corporation. Sun is a registered trademark of Sun Microsystems Inc. All other trademarks mentioned belong to their respective owners.

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 5
1.1 Purpose and audience 5
1.2 Summary of content 5

2 New features, parameters and corrections 7
2.1 New features 7
2.2 New parameters 7
2.3 Deleted parameters 7
2.4 Changed parameter names 8
2.5 Changed (renamed) parameter values 8
2.6 Converting “old” parameters to “new” with “we-pconv”. 8

3 New installation procedure 11
3.1 Extracting the product from the distribution media 11
3.2 What to do when extraction is complete 11
3.2.1 Reading the README file in the installation directory 11
3.2.2 Running the product with the Sample configuration file 11

4 Session logging fro WE-D320 13
4.1 Overview 13
4.2 Preparing WE-D320 for logging 13
4.3 **Mapping the “M_XXX_LOGGING” functions to the keyboard 13**
4.4 **Mapping the “M_XXX_LOGGING” functions to the buttons 14**
4.5 **How the operator can see when logging is active 14**
4.6 **Where the logged data will go 14**

5 The WE-HLLAPI programming interface 17
5.1 Foreword 17
5.2 Overview 17
5.3 WE-HLLAPI application overview 18
5.4 New parameter for the WE-XXX emulation product 18
5.5 Important remarks 18
5.6 library calls, hllc functions, attributes and send key mnemonics 19

6 The WE-SCRIPT script language 25
6.1 Foreword 25
6.2 When should one use WE-SCRIPT or WE-HLLAPI 25
6.3 4. WE-SCRIPT variables. 25
6.4 Invoking WE-SCRIPT 26
6.5 WE-SCRIPT token list 26
6.6 Using WE-SCRIPT, guidelines 27
6.7 A commented WE-SCRIPT example 28

7 Changes to the Keymapper tool 31
7.1 Overview 31
7.2 The new <Keyboard_Kind> emulator option. 31
7.3 Keyboard Layout show mode . 32
7.4 New functions mappable to the emulator keyboard or buttons. 32

7.5 Miscellaneous concerning key mapping tool and emulator. 32

8 The WE-LICD licence server program 35
8.1 Overview 35
8.2 Purpose of the password 35
8.3 The password file 35
8.4 New parameter for the emulation products using WE-LICD 36
8.5 Running WE-LICD 36

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-D320 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-D320 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-D320.

Chapter 4: "Session logging feature"
explains how session logging for WE-D320 works.

Chapter 5: "The WE-HLLAPI programming interface"
describes the implementation of this API within WE-D320.

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-D320 and provides some examples how it may be used to make an automatic session logon.

Chapter 7: "Changes for the Keymapper tool"
explains the new features of the KM-D320 keyboard mapping tool.

Chapter 8: "The WE-LICD licence server program"
explains how to setup and run the floating licence server for WE-D320 and how it interacts with WE-D320.

2 New features, parameters and corrections

2.1 New features

Since Release 1.633, the WE-D320 product has been expanded with the following important new features which will be described in more details in this manual:

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

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

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

-> Session logging (both commanded from host and from operator) implemented.

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

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

2.2 New parameters

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

->**Keyboard_Kind**	You may set values (currently from 1 to 5). See chapter 8 for more informations..
-> **API_Service**	Is used while you intend to start WE-D320 with an WE-HLLAPI interface. See WE-HLLAPI and WE-SCRIPT chapters 5 & 6.
-> **Host_License_Server**	Is used if you want to get the WE-D320 licence from the WE-LICD floating licence server. See WE-LICD licence program chapter 8.

2.3 Deleted parameters

A brief list of deletd parameters follows:

-> **Font_R2**	No more useful
-> **Font_B2**	No more useful
-> **Font_File**	No more useful
-> **Column_132**	No more useful

Such entries will be commented out if you update you config files with “we-pconv” (see below).

2.4 Changed parameter names

A list of renamed parameters follows:

-> **Font_R1**	Replaced by:	**Font_R**
-> **Font_B1**	Replaced by:	**Font_B**
-> **Font_Panel**	Replaced by:	**Font_P**

These parameter names have been changed for compatibility with other members (WE-I3179, WE-UTS) of the WE-XXX terminal emulation products family, thus allowing sharing more common code between the emulations. For the customer, sharing more code means that corrections/improvements made in one product flows quicker to the whole product range.

2.5 Changed (renamed) parameter values

A brief list of renamed parameter values follows:

-> **Keypad_Mode Normal**	Replaced by:	**Keypad_Mode Numeric**
-> **Cursor_Keys Numeric**	Replaced by:	**Cursor_Keys Cursor**

These 2 changes have been made to give more logical (and DEC like) names to the above 2 parameter values.

2.6 Converting “old” parameters to “new” with “we-pconv”.

In order to help customers converting their “old” config files to the new parameters names/values, we provide the “we-pconv” utility located in the “WE-PCONV”sub directory in the installation directory (or in the .app for NeXTStep).

“we-pconv” is a shell script that you can start as follow:

we-pconv ConfigFileName

“we-pconv” will make the following changes in “ConfigFileName”:

-> Replace Font_Panel>	with <Font_P>,'
-> Replace Font_R1>	with <Font_R>,'
-> Replace Font_B1>	with <Font_B>,'
-> Replace Font_R2>	with <#Font_R2>,'
-> Replace Font_B2>	with <#Font_B2>,'
-> Replace Keypad_Mode Normal>	with <Keypad_Mode Numeric>,'
-> Replace Cursor_Keys Numeric>	with <Cursor_Keys Cursor>,'

-> Replace Font_File>	<#Font_File>., thus commenting it
-> Replace Column_132>	<#Column_132>, thus commenting it

If you integrate “we-pconv” in some other shell script searching for WE-D320 config files and then automatically executing “we-pconv” on each of them, you may update your config files quite quickly.

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 “csh” (the C shell) is started wihin WE-D320. This allows to make some experiences using WE-D320 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.

4 Session logging fro WE-D320

4.1 Overview

-> Session logging enables users to store the dialog between WE-D320 and the application in a file or to print it.

-> Session logging merely logs/prints the incoming characters without any interpretation. This means that any attributes and/or cursor control sequences will be logged/printed as is though they are not useful data.

4.2 Preparing WE-D320 for logging

Session logging may be enabled/disabled by the operator and by the host.

=> The host may start/stop logging using the following DEC defined escape sequences:

CSI ? 5 i	Will start logging (DEC calls this Autoprint Mode)
CSI ? 4 i	Will stop logging

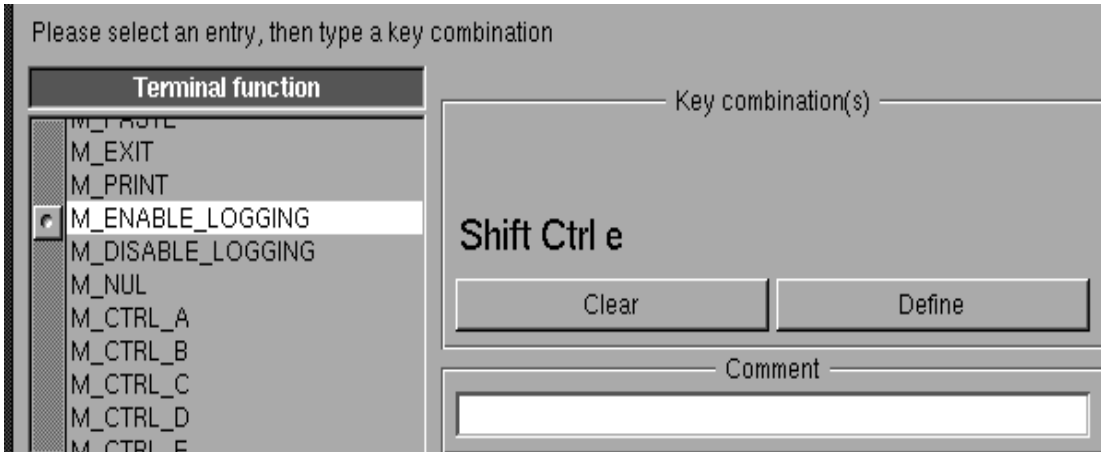
=> The operator starts/stops logging using:

M_ENABLE_LOGGING	Will start logging
M_DISABLE_LOGGING	Will stop logging

The host functions are already “mapped” within the WE-D320 and needs no configuration. The mapping of the M_XXX_LOGGING functions may be done with the keyboard mapping tool, the buttons definition file or both.

4.3 Mapping the “M_XXX_LOGGING” functions to the keyboard

After starting km-d320 with your particular keyboard mapping file (see keyboard mapping chapter), assign keystrokes to the M_ XXX_LOGGING functions as follow:



When this is done, save you keyboard mapping file and either restart WE-D320 or reread the keyboard mapping by clicking the “read again” button on the parameter panel. You will then be able to start and stop logging using the assigned keystrokes.

4.4 Mapping the “M_XXX_LOGGING” functions to the buttons

You may assign any “M_” function to the buttons surrounding your WE-D320 window. The 2 M_XX_LOGGING functions are good candidates for such mapping. You may insert lines such as the following in you buttons mapping file:

< 1, "BLOG", M_ENABLE_LOGGING >	Button 1 will be labeled "BLOG" and will start logging
< 2, "ELOG", M_DISABLE_LOGGING >	Button 2 will be labeled "ELOG" and will end logging

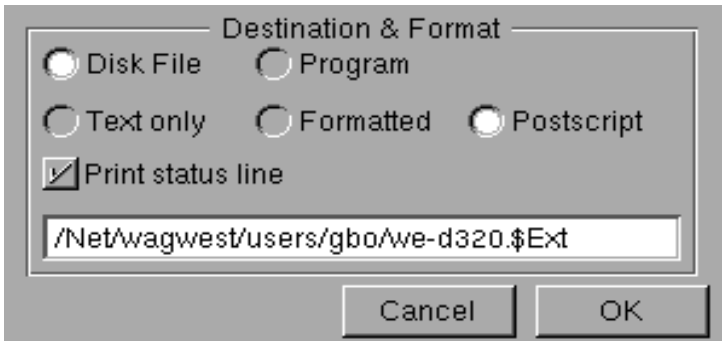
4.5 How the operator can see when logging is active

The logging activity is displayed in the same field as the printer activity. While logging is active, the “print” field on the status line will look as follow:

4.6 Where the logged data will go

Logged data will go either to a file or be printed when you stop the logging mode. The choice between these two is done with the Print Format menu. This means that the entries made in this menu (and the defaults read from the configuration file) are valid for both printing and logging.

=> If you want your logging data to go on disk, you may enter the following in the Print Format menu (this may also be default values coming from your config file):



This means that logging data will be stored in the following file names (1 file for each logging start/stop cycle):

-> The first file will be: /Net/wagwest/users/gbo/we-d320.PID-0-logging

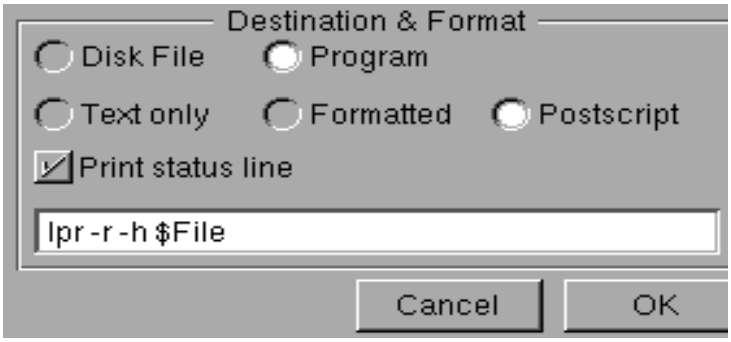
-> The second file will be: /Net/wagwest/users/gbo/we-d320.PID-1-logging

-> The n'th file will be: /Net/wagwest/users/gbo/we-d320.PID-n-logging

Where PID is the process ID of the particular WE-D320 session.

=> If you want your logging data to be printed, you may enter the following in the Print Format menu (this may also be default values coming from your config file)

:



This will print the logged data on the default printer.

NOTES:

-> The format (Text only, Formatted or Postscript) is used only for hardcopies (not for logging). Logging is alway done in Text only mode.

-> If you make both logging and hardcopies to files, we strongly recommend that you append the $EXT extension at the end of your filename in the Print Format menu. This will instruct WE-D320 to replace it with the "hcopy" extension for hardcopies and with the "logging" extension for logging and enable autonumbering the files (see above). Doing so, you will have no problem to retrieve your data.

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

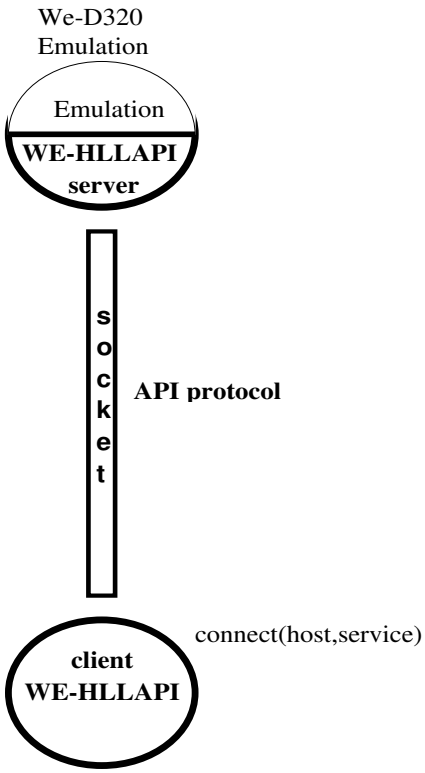
-> WE-HLLAPI is designed to give users and programmers access to the content of the emulator window (sometimes named the host presentation space) with a set of functions that can be called from an application program. Because the emulation handles all network communications to the host system, the user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

7.1 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.

7.2 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-d320 NeXT **Us.D320_keymap** **-kk 4**

This line would allow one to setup a good keyboard mapping for the WE-D320 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**

7.3 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...)

7.4 New functions mappable to the emulator keyboard or buttons.

The following mappable functions have been added to WE-D320 and can be found in the list displayed by the KM-D320 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

7.5 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, ...).

-> 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.

Changes to the Keymapper 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.

8 The WE-LICD licence server program

8.1 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.

8.2 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

8.3 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.

8.4 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.

8.5 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.