Emulator Programming

EHLLAPI Functions

This chapter describes each individual Personal Communications EHLLAPI function in detail and explains how to use the EHLLAPI program sampler. The functions are arranged alphabetically by name. The functions are explained for both the standard and enhanced interfaces.

Note:

Throughout this chapter WinHLLAPI, IBM Standard 32-bit HLLAPI and 16-bit EHLLAPI are referred to as Standard Interface, and IBM Enhanced 32-bit EHLLAPI is referred to as Enhanced Interface.

Unicode Support for Code Pages 1390/1399 and 1137

The following EHLLAPI functions are enabled for Japanese code page 1390/1399 and Hindi code page 1137 support on a Unicode session:

- Convert Position or Convert RowCol (1137 only)
- Copy Field to String
- Copy Presentation Space
- Copy Presentation Space to String
- Copy String to Field
- Copy String to Presentation Space
- Get Key
- Search Field
- Search Presentation Space
- Send Key
- Set Cursor (1137 only)
- Set Session Parameters

See the specific section for each function for details on Japanese code page 1390/1399 and Hindi code page 1137.

Notes:

1. The string containing the Unicode characters to be sent to the PCOMM session should be typecast to `WCHAR *` for code page 1390/1399 and to `char *` for code page 1137.
2. EHLLAPI 1390/1399 Unicode functionality is available only for 3270 and 5250 sessions. EHLLAPI 1137 Unicode functionality is available only for 5250 sessions.

Page Layout Conventions

All EHLLAPI function calls are presented in the same format so that you can quickly retrieve the information you need. The format is:

Function Name (Function Number)

Prerequisite Calls

Call Parameters

Return Parameters

Notes on Using This Function

Prerequisite Calls

"Prerequisite Calls" lists any calls that must be made prior to calling the function being discussed.

Call Parameters

"Call Parameters" lists the parameters that must be defined in your program to call the discussed EHLLAPI function and explains how those parameters are to be defined. If a parameter is never used by a function, then *NA* (not applicable) is listed. If a parameter can be overridden by certain values of session parameters defined with calls to the **Set Session Parameters** (9) function, such session parameters are named.

Return Parameters

"Return Parameters" lists the parameters that must be received by your program after a call to the discussed EHLLAPI function and explains how to interpret those parameters.

Notes on Using This Function

"Notes on Using This Function" lists any session options that affect the function under discussion. It also provides technical information about using the function and application development tips.

Summary of EHLLAPI Functions

Table 6 is the summary of the EHLLAPI functions:

Table 6. EHLLAPI Functions Summary

Function	3270	5250	VT
Connect Presentation Space (1)	Yes	Yes	Yes
Disconnect Presentation Space (2)	Yes	Yes	Yes
Send Key (3)	Yes	Yes	Yes
Wait (4)	Yes	Yes	Yes
Copy Presentation Space (5)	Yes	Yes	Yes
Search Presentation Space (6)	Yes	Yes	Yes
Query Cursor Location (7)	Yes	Yes	Yes
Copy Presentation Space to String (8)	Yes	Yes	Yes
Set Session Parameters (9)	Yes	Yes	Yes
Query Sessions (10)	Yes	Yes	Yes

Reserve (11)	Yes	Yes	Yes
Release (12)	Yes	Yes	Yes
Copy OIA (13)	Yes	Yes	Yes
Query Field Attribute (14)	Yes	Yes	Yes
Copy String to Presentation Space (15)	Yes	Yes	Yes
Pause (18)	Yes	Yes	Yes
Query System (20)	Yes	Yes	Yes
Reset System (21)	Yes	Yes	Yes
Query Session Status (22)	Yes	Yes	Yes
Start Host Notification (23)	Yes	Yes	Yes
Query Host Update (24)	Yes	Yes	Yes
Stop Host Notification (25)	Yes	Yes	Yes
Search Field (30)	Yes	Yes	Yes
Find Field Position (31)	Yes	Yes	Yes
Find Field Length (32)	Yes	Yes	Yes
Copy String to Field (33)	Yes	Yes	Yes
Copy Field to String (34)	Yes	Yes	Yes
Set Cursor (40)	Yes	Yes	Yes
Start Close Intercept (41)	Yes	Yes	Yes
Query Close Intercept (42)	Yes	Yes	Yes
Stop Close Intercept (43)	Yes	Yes	Yes
Query Additional Field Attribute DRB	No	Yes	No
Start Keystroke Intercept (50)	Yes	Yes	Yes
Get Key (51)	Yes	Yes	Yes
Post Intercept Status (52)	Yes	Yes	Yes
Stop Keystroke Intercept (53)	Yes	Yes	Yes
Lock Presentation Space API (60)	Yes	No	No
Lock Window Services API (61)	Yes	No	No
Start Communication Notification (80)	Yes	Yes	Yes
Query Communication Event (81)	Yes	Yes	Yes
Stop Communication Notification (82)	Yes	Yes	Yes
Send File (90)	Yes	Yes	No
Receive File (91)	Yes	Yes	No
Cancel File Transfer (92)	Yes	Yes	Yes
Convert Position or Convert RowCol (99)	Yes	Yes	Yes
Connect Window Services (101)	Yes	Yes	Yes
Disconnect Window Service (102)	Yes	Yes	Yes
Query Window Coordinates (103)	Yes	Yes	Yes
Window Status (104)	Yes	Yes	Yes
Change Switch List LT Name (105)	Yes	Yes	Yes
Change PS Window Name (106)	Yes	Yes	Yes
Start Playing Macro (110)	Yes	Yes	Yes
Connect for Structured Fields (120)	Yes	No	No
Disconnect from Structured Fields (121)	Yes	No	No
Query Communications Buffer Size (122)	Yes	No	No
Allocate Communications Buffer (123)	Yes	No	No
Free Communications Buffer (124)	Yes	No	No
Get Request Completion (125)	Yes	No	No
Read Structured Fields (126)	Yes	No	No
Write Structured Fields (127)	Yes	No	No

Allocate Communications Buffer (123)

3270	*5250*	*VT*
Yes	No	No

The **Allocate Communications Buffer** function obtains a buffer from the operating system. A buffer address must be passed on both the **Read Structured Fields** (126) and **Write Structured Fields** (127) functions.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 123	
Data String	See the following table	
Length	Must be 6	Must be 8
PS Position	NA	

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1-2	1-4	32-bit or 16-bit buffer length. (0 < size <= (64 KB-256 bytes)=X'FF00')
3-6	5-8	32-bit allocated buffer address (returned)

Return Parameters

Return Code	Explanation
0	The **Allocate Communications Buffer** function was successful.
2	An error was made in specifying parameters.
9	A system error occurred.
11	Resource unavailable (memory unavailable).

Notes on Using This Function

1. The EHLLAPI obtains a buffer from the operating system memory management and places the buffer address into the return parameter string. The requested buffer size (length) is also passed in the parameter string. The buffer size can be from 1 byte to 64 KB minus 256 bytes (X'FF00' bytes) in length.

   See "**Query Communications Buffer Size** (122)" for information regarding buffer size.

2. Buffers obtained using this function must not be shared among different processes. If this is attempted, the applications will experience unpredictable results.
3. An EHLLAPI application must issue a **Free Communications Buffer** (124) function to free the allocated memory.
4. A maximum of 10 buffers can be allocated to an application. If this limit is reached, a return code for resource unavailable (RC=11) will be returned.
5. The **Reset System** (21) function frees buffers allocated by this function.

Cancel File Transfer (92)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Cancel File Transfer** function causes any current EHLLAPI initiated **Send File** or **Receive File** for the specified session to immediately return.

Prerequisite Calls

Send File (90) or **Receive File** (91)

Call Parameters

	Enhanced Interface
Function Number	Must be 92
Data String	1-character short name of the host presentation space. A blank or null indicates request for updates to the host-connected presentation space
Length	4 is implied
PS Position	NA

The calling data structure contains these elements

Byte	Definition
1	A 1-character presentation space short name (PSID)
2-4	Reserved

Return Parameters

Return Code	Definition
0	The function was successful
1	An incorrect PSID was specified
8	No prior call to **Start Communication Notification** (80) function was called for the PSID
9	A system error was encountered

Notes on Using This Function

Since both **Send File** (90) and **Receive File** (91) are blocking calls, this function must always be issued on a different thread.

Change PS Window Name (106)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Change PS Window Name** function allows the application to specify a new name for the presentation space window or reset the presentation space window to the default name.

Prerequisite Calls

Connect Window Services (101)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 106	
Data String	See the following table	
Length	Must be specified (See note.)	Must be 68
PS Position	NA	

Note:

The data string length must be specified (normally 3-63 for PC/3270, 4-63 for PC400, 68 for enhanced interface).

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-4	Reserved
2	5	A change request option value, select one of: • X'01' for changing the presentation space window name. • X'02' for resetting the presentation space window name.
3-63	6-66	An ASCII string of from 1 (for PC/3270) or 2 (for PC400) to 61 bytes including a terminator byte. The ASCII string must end with a NULL character. This string must contain at least one non-NULL character followed by a NULL character.
	67-68	Reserved

Return Parameters

Return Code	Explanation
0	The **Change PS Window Name** function was successful.
1	An incorrect host presentation space short session ID was specified, or the host presentation space was not connected.
2	An error was made in specifying parameters.
9	A system error occurred.
12	The session stopped.

Notes on Using This Function

A string is ended at the first NULL character found. The NULL character overrides the specified string length. If the NULL character is not at the end of the specified length, the last byte at the specified length is replaced by a NULL character, and the remainder of the data string is lost. If the NULL character is found before the specified length, the string is truncated at that point, and the remainder of the data string is lost.

If the application fails to reset the presentation space name before exiting, the exit list processing resets the name.

Change Switch List LT Name (105)

3270	*5250*	*VT*

Yes	Yes	Yes

The **Change Switch List LT Name** function allows the application to change or reset a switch list for a selected logical terminal (LT). The application must specify on the call the name to be inserted in the switch list.

Note:

This is for compatibility with Communication Manager EHLLAPI, and has the same result as the **Change PS Window Name** (106) function.

Prerequisite Calls

Connect Window Services (101)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 105	
Data String	See the following table	
Length	Normally 4-63	Must be 68
PS Position	NA	

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-4	Reserved
2	5	A change request option; select: • X'01' for changing a switch list LT name • X'02' for resetting a switch list LT name
3-63	6-66	An ASCII string of 2 to 61 bytes including a terminator byte. The ASCII string must end with a NULL character. This string must contain at least one non-NULL character followed by a NULL character.
	67-68	Reserved

Return Parameters

Return Code	Explanation
0	The **Change Switch List LT Name** function was successful.
1	An incorrect host presentation space short session ID was specified, or the host presentation space was not connected.
2	An error was made in specifying parameters.
9	A system error occurred.
12	The session stopped.

Notes on Using This Function

A string is ended at the first NULL character found. The NULL character overrides the specified string length. If the NULL character is not at the end of the specified length, the last byte at the specified length is replaced by a NULL character, and the remainder of the data string is lost. If the NULL character is found before the specified length, the string is truncated at that point, and the remainder of the data string is lost.

If the application fails to reset the switch list LT name before exiting, the exit list processing resets the name.

Connect for Structured Fields (120)

3270	*5250*	*VT*
Yes	No	No

The **Connect for Structured Fields** function allows an application to establish a connection to the emulation program to exchange structured field data with a host application. The workstation application must provide the Query Reply data field and must point to it with in the parameter string. The destination/origin ID returned by the emulator will be returned to the application.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 120	
Data String	See the following table	
Length	7 or 11	Must be 16
PS Position	NA	

The calling data string can contain:

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-4	Reserved
2-5	5-8	Address of the Query Reply data buffer
6-7	9-10	Destination/origin unique ID. (16-bit word, returned)
	11-12	Reserved
8-11	13-16	The data in these position is ignored by EHLLAPI. However, no error is caused if the migrating program has data in these positions. This data is accepted to provide compatibility with migrating applications.

Return Parameters

Return Code	**Explanation**
0	The **Connect for Structured Fields** function was successful.
1	A specified host presentation space short session ID was not valid, or the host presentation space was not connected.
2	An error was made in specifying parameters.
9	A system error occurred.
10	The function is not supported by the emulation program.
32	An application has already connected to this session for communications (successful connect).
39	One DDM session is already connected to this session.

Notes on Using This Function

1. EHLLAPI scans the query reply buffers for the destination/origin ID (DOID) self-defining parameter (SDP) to determine the contents of the DOID field of the query reply. If this value is X'0000', the emulator will assign a DOID to the application and EHLLAPI will fill in the DOID field of the query reply with the assigned ID. If the value specified by the application in the DOID field of the query reply is a nonzero value, the emulator will assign the specified value as the application's DOID, assuming that the ID has not been previously assigned. If the specified DOID is already in use, a return code of 2 will be returned by EHLLAPI.
2. The application should build the Query Reply Data structures in the application's private memory. Refer to Appendix A, Query Reply Data Structures Supported by EHLLAPI, for the detailed formats and usages of the query reply data structures supported by EHLLAPI.
3. Only cursory checking is performed on the Query Reply Data. Only the ID and the length of the structure are checked for validity.
4. Only one DDM base type connect is allowed per host session. If the DDM connection supports the self-defining parameter (SDP) for the destination origin ID (DOID), then multiple connects are allowed.
5. If return code RC=32 or RC=39 is received, an application is already connected to the selected session and use of that presentation space should be approached with caution. Conflicts with SRPI, file transfer, and other EHLLAPI applications might result.

Connect Presentation Space (1)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Connect Presentation Space** function establishes a connection between your EHLLAPI application program and the host presentation space.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 1	
Data String	1-character short name of the host presentation space	
Length	1 is implied	Must be 4
PS Position	NA	

The calling data string can contain:

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-4	Reserved

Return Parameters

The **Connect Presentation Space** function sets the return code to indicate the status of the attempt and, if successful, the status of the host presentation space.

Return Code	Explanation
0	The **Connect Presentation Space** function was successful; the host presentation space is unlocked and ready for input.
1	An incorrect host presentation space ID was specified. The specified session either does not exist or is a logical printer session. This return code could also mean that the API Setting for DDE/EHLLAPI is not set on.
4	Successful connection was achieved, but the host presentation space is busy.

5	Successful connection was achieved, but the host presentation space is locked (input inhibited).
9	A system error was encountered.
11	This resource is unavailable. The host presentation space is already being used by another system function.

Notes on Using This Function

1. The **Connect Presentation Space** function is affected by the `CONLOG`/`CONPHYS` session option.
2. An EHLLAPI application cannot be connected to multiple presentation spaces concurrently. Calls requiring the **Connect Presentation Space** function as a prerequisite use the currently connected presentation space. For example, if an application is connected to presentation space A, B, and C in that order, the application must connect to B or A again to issue functions.
3. Each thread that requests a **Connect Presentation Space** must have a corresponding **Disconnect Presentation Space** (2), or one of the threads must issue a **Reset System** (21), which affects all threads and disconnects any remaining connections.
4. More than one EHLLAPI application can share a presentation space, if the applications support sharing (that is, if they were developed to work together and if they exhibit predictable behavior) and have compatible read/write access and keyword options as set in the **Set Sessions Parameters** (9) function. For more information, see Set Session Parameters (9).
5. Because the **Connect Presentation Space** and **Start Keystroke Intercept** (50) functions share common subsystem functions, successful requests by an application to share either of these functions for the same session can affect the request of these two functions by other applications. For example, if application A successfully requests a **Connect Presentation Space** for a session with Write_Read access and KEY$abcdefgh as the keyword, a request by application B to **Connect Presentation Space** for a session and **Start Keystroke Intercept** is successful only if both applications have set compatible read/write options.
6. You cannot connect to a session that is defined as a logical printer session. Refer to *Administrator's Guide and Reference* for more information.

Connect Window Services (101)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Connect Window Services** function allows the application to manage the presentation space windows. Only one EHLLAPI application at a time can be connected to a presentation space for window services.

An EHLLAPI application can connect to more than one presentation space concurrently for window services.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 101	
Data String	1-character short session ID of the host presentation space	
Length	1 is implied	Must be 4
PS Position	NA	

The calling data string can contain:

Byte		**Definition**
Standard	Enhanced	

1	1	A 1-character presentation space short name (PSID)
	2-4	Reserved

Return Parameters

Return Code	Explanation
0	The **Connect Window Services** function was successful.
1	An incorrect host presentation space short session ID was specified, or the Sessions Window Services manager was not connected. This return code could also mean that the API Setting for DDE/EHLLAPI is not set on.
9	A system error occurred.
10	The function is not supported by the emulation program.
11	This resource is unavailable. The host presentation space is already being used by another system function.

Notes on Using This Function

1. An EHLLAPI application can be connected to multiple presentation space windows at the same time. The application can go back and forth between the connected presentation space windows without having to disconnect. For example, if an application is connected to presentation space windows A, B, and C, the application can access all of A, B, and C at the same time, and the other applications cannot access A, B, or C.
2. A **Connect Window Services** function is sufficient for the process. However, each thread that requests a **Connect Window Services** must have a corresponding **Disconnect Window Services** (102), or one of the threads must issue a **Reset System** (21), which affects all threads and disconnects any remaining connections.

Convert Position or Convert RowCol (99)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Convert Position** or **Convert RowCol** function converts the host presentation space positional value into the display row and column coordinates or converts the display row and column coordinates into the host presentation space positional value. This function does not change the cursor position.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 99	
Data String	Host presentation space short name *and* `P` for the **Convert Position** function (for example, `AP` converts the presentation space position of session A); *or* Host presentation space short name and `R` for the **Convert RowCol** function (for example, `AR` converts the row and column coordinates of session A).	
Length	Row, when `R` is specified as the second character in the data string parameter. The lower limit for valid input is 1. The upper limit for valid input depends on how your host presentation space is configured. See Notes on Using This Function. NA when `P` is specified as the second character in the data string parameter.	
PS Position	Column, when `R` is specified as the second character in the data string parameter. The lower limit for valid input is 1. The upper limit for valid input ranges from 24 to 43 depending on how your host presentation space is configured. See Notes on Using This Function.	

	Host presentation space position, when `P` is specified as the second character in the data string parameter. The lower limit for valid input is 1. The upper limit for valid input ranges from 1920 to 3564 depending on how your host presentation space is configured. See Notes on Using This Function.

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-4	Reserved
2	5	**Convert** option `P` or `R`
	6-8	Reserved

Return Parameters

This function returns a length and a return code.

Length:

For the **Convert Position** function (`P` as the second character in the calling data string), a number between 1 and 43 (for PC/3270) or 27 (for PC400) is returned. This value is the number of the row that contains the PS position contained in the calling PS position parameter. The upper limit can be smaller than 43 (for PC/3270) or 27 (for PC400) depending on how the host presentation space is configured.

For the **Convert RowCol** function (`R` as the second character in the calling data string), a value of 0 indicates an error in the input value for row (calling length parameter).

Return Code:

The **Convert Position or RowCol** function is the exception to the rule that the fourth return parameter always contains a return code. For this function, the value returned in the fourth parameter is called a status code. This status code can contain data or a return code. Your application must provide for processing of this status code to prevent unpredictable results or an error.

- If the value of the fourth parameter is 0, 9998, or 9999, it is a return code.
- For the **Convert Position** function (`P` as the second character of the calling data string), a value in the range of 1-132 is the number of the column that contains the PS position passed in the calling PS Position parameter. The upper limit can be smaller than 132 depending on how the host presentation space is configured.
- For the **Convert RowCol** function (`R` as the second character of the calling data string), a value in the range of 1-3564 represents the host presentation space position that corresponds to the row and column values passed in the calling length and PS position parameters, respectively. The upper limit can be smaller than 3564 depending on how the host presentation space is configured.

The following status codes are defined:

Status Code	Explanation
0	This is an incorrect PS position or column.
>0	This is the PS position or column.
9998	An incorrect host presentation space ID was specified or a system error occurred.
9999	Character 2 in the data string is not P or R.

Notes on Using This Function

1. To configure your presentation space, refer to *Administrator's Guide and Reference*
2. To find out how many rows and columns are in your presentation space, examine the returned data string parameter for the **Query Session Status** (22) function. See Query Session Status (22).

1137 Code Page Support

Unicode functionality is supported only on 5250 sessions.

Convert Position or Convert RowCol is Hindi enabled in order to return the beginning of the cluster. The usage of **Convert Position or Convert RowCol** is the same as the SBCS session.

Copy Field to String (34)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Copy Field to String** function transfers characters from a field in the host-connected presentation space into a string.

The **Copy Field to String** function translates the characters in the host source presentation space into American National Standard Code for Information Interchange (ASCII). Attribute bytes and other characters not represented in ASCII normally are translated into blanks.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 34	
Data String	Preallocated target data string. When the **Set Session Parameters** (9) function with Extended Attribute Bytes (EAB) option is issued, the length of the data string must be at least twice the length of the field. **DBCS Only:** When Extended Attributes Double-byte (EAD) option is specified, the length of the data string must be at least three times the length of the field. When both EAB and EAD options are specified, the length of the data string must be at least four times the length of the field.	
Length	Number of bytes to copy (the length of the data string).	
PS Position	Identifies the target field. This can be the PS position of any byte within the target field. Copy always starts at the beginning of the field.	

Return Parameters

This function returns a data string, length, and a return code.

Data String:

A string containing data from the identified field in the host presentation space. The first byte in the returned data string is the beginning byte of the identified field in the host presentation space. The number of bytes in the returned data string is determined by the smaller of:

- Number of bytes specified in the calling length parameter
- Number of bytes in the identified field in the host presentation space

Length:

The length of the data returned.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Copy Field to String** function was successful.
1	Your program is not connected to a host session.
2	An error was made in specifying parameters.
6	The data to be copied and the target field are not the same size. The data is truncated if the string length is smaller than the field copied.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	Unformatted host presentation space.

Notes on Using This Function

1. The field position and length information can be found by using the **Find Field Position** (31) and **Find Field Length** (32) functions. The **Copy Field to String** function can be used with either protected or unprotected fields, but only in a *field-formatted* host presentation space.

2. The copy is ended when one of the following conditions is encountered:
   - When the end of the field is reached
   - When the length of the target string is exceeded
3. **DBCS Only:** If the target string is ended at the higher byte of the DBCS character, the byte is translated into a blank. If the EAD option is set to on, three bytes are returned for each character. If both the EAB and EAD options are set to on, four bytes are returned for each character.

   Note:

   When the field wraps at the end of the presentation space, wrapping occurs when the end of the presentation space is reached.

4. **DBCS Only:** The **Set Session Parameters** (9) function EAD option is used with this function to return a 2-byte EAD. If the EAD option is specified instead of the EAB option, EAD is returned preceding each character. If both the EAB and EAD options are specified, EAD is returned preceding the EAB.
5. An EAB can be returned when the **Set Session Parameters** (9) function EAB option is used. EAB is related to each character in the presentation space and is returned preceding each character.
6. The **Copy Field to String** function is affected by the `ATTRB`/`NOATTRB`/`NULLATTRB`, the `EAB`/`NOEAB`, the `XLATE`/`NOXLATE`, the `DISPLAY`/`NODISPLAY`, the `DISPLAY`/`NODISPLAY`, the `EAD`/`NOEAD` (for DBCS only), and the `NOSO`/`SPACESO`/`SO` (for DBCS only) session options. Refer to items 5; 13 and 14; 17; and 20 and 21 for more information.

   As previously stated, the return of attributes by the various **Copy** (5, 8, and 34) functions is affected by the **Set Session Parameters** (9) function. The involved set session parameters have the following effect:

   Set Session Parameter

   Effect on the COPY Function

   NOEAB and NOEAD

Attributes are not returned. Only text is copied from the presentation space to the user buffer.

EAB and NOXLATE

Attributes are returned as defined in the following tables.

EAB and XLATE

The colors used for the presentation space display are returned. Colors can be remapped; so the attribute colors are not the ones returned by the **COPY** functions when XLATE and EAB are on at the same time.

EAD

Double-byte character set attributes are returned as shown in the following tables.

The returned character attributes are defined in the following tables. The attribute bit positions are in IBM format with bit 0 the left most bit in the byte.

3270 character attributes are returned from the host to the emulator. The following table applies when EAB and NOXLATE are set.

Bit Position	Meaning
0-1	Character highlighting 00 = Normal 01 = Blink 10 = Reverse video 11 = Underline
2-4	Character color (Color remap can override this color definition.) 000 = Default 001 = Blue 010 = Red 011 = Pink 100 = Green 101 = Turquoise 110 = Yellow 111 = White
5-6	Character attributes 00 = Default value 11 = Double byte character
7	Reserved

5250 character attributes are returned from the host to the emulator. The following table applies when EAB and NOXLATE are set.

Bit Position	Meaning
0	Reverse image

	0 = Normal image 1 = Reverse image
1	Underline 0 = No underline 1 = Underline
2	Blink 0 = Not blink 1 = Blink
3	Separator of columns 0 = No separator 1 = Separator
4-7	Reserved

The following table shows Personal Communications character color attributes. The following table applies when EAB and XLATE are set.

Bit Position	Meaning
0-3	Background character colors 0000 = Black 0001 = Blue 0010 = Green 0011 = Cyan 0100 = Red 0101 = Magenta 0110 = Brown (3270), Yellow (5250) 0111 = White
4-7	Foreground character colors 0000 = Black 0001 = Blue 0010 = Green 0011 = Cyan 0100 = Red 0101 = Magenta 0110 = Brown (3270), Yellow (5250) 0111 = White 1000 = Gray 1001 = Light blue

	1010 = Light green 1011 = Light cyan 1100 = Light red 1101 = Light magenta 1110 = Yellow 1111 = White (high intensity)

- Double-byte character set attributes (for DBCS only)
  - The first byte

Bit Position	Character Position	Field Attribute Position
0	Double-byte character	Reserved
1	The first byte of the double-byte character	Reserved
2	SO	Reserved
3-4	SI (Bit position 3)	5250 DBCS related field When the value of bit position 7 is 0: 00 = Default 01 = DBCS only 10 = Either DBCS or SBCS 11 = Mixture of DBCS and SBCS When the value of bit position 7 is 1: 00 = Reserved 01 = DBCS only without SO/SI 10 = Reserved 11 = Reserved
5	Reserved	SO/SI enable (3270 only)
6	Reserved	Character attributes exist (3270 only)
7	Reserved	5250 DBCS related extended field 0 = Basic double-byte field 1 = Extended double-byte field

  - The second byte

Bit Position	Character Position	Field Attribute Position
0	Reserved	Left grid line (3270 only)
1	Reserved	Upper grid line (3270 only)
2	Reserved	Right grid line (3270 only)
3	Reserved	Under grid line (3270 only)
4	Left grid line	Left grid line
5	Upper grid line	Upper grid line
6-7	Reserved	Reserved

For a PS/2[(R)] monochrome display, the characters in the application (workstation) session appear as various shades of gray. This is required to give users their remapped colors in the EHLLAPI application session so they can get what they see in their host application presentation spaces.

7. To use this function, preallocate memory to receive the returned data string parameter. The statements required to preallocate this memory vary depending on the language in which your application is written. Refer to Memory Allocation for more information.

Note:

5250 emulation supports a presentation space of 24 rows by 80 columns. In some instances, Communication Manager 5250 emulation displays a 25th row. This occurs when either an error message from the host is displayed or when the operator selects the SysReq key. Personal Communications displays 25th row information on the status bar. By **EXTEND_PS** option, an EHLLAPI application can use the same interface with Communication Manager EHLLAPI and valid presentation space is extended when this condition occurs.

1390/1399 Code Page Support

Unicode functionality is supported only on 3270 and 5250 sessions.

In a Unicode session, the characters in the host source presentation space are translated into Unicode. Attribute bytes are normally translated into blanks.

The XLATE option (that can be specified using the **Set Session Parameters** function) is not supported in a Unicode session. This means that even if this option is issued, the EABs will not be translated to the PC color graphics adapter (CGA) format.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 34	
Data String	Preallocated target data string. When the **Set Session Parameters** (9) function with Extended Attribute Bytes (EAB) option is issued, the length of the data string must be at least twice the length of the EBCDIC field.	
Length	The length of the target data string in Unicode characters.	
PS Position	Identifies the target field. This can be the PS position of any byte within the target field. Copy always starts at the beginning of the field.	

Return Parameters

This function returns a data string, length, and a return code.

Data String:

String containing the Unicode data is returned.

Length:

Number of Unicode characters copied into string.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Copy Field to String** function was successful.
1	Your program is not connected to a host session.
2	An error was made in specifying parameters.
6	The data to be copied and the target field are not the same size. The data is truncated if the string length is smaller than the field copied.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	Unformatted host presentation space.

Notes on Using This Function

The following options are supported in a Unicode session for **Copy Field To String** (34) and function in the same way as in DBCS:

- NOATTRB
- ATTRB
- NULLATTRB
- EAB
- NOEAB
- NOXLATE
- DISPLAY
- NODISPLAY

1137 Code Page Support

Unicode functionality is supported only on 5250 sessions.

In a Unicode session, the characters in the host source presentation space are translated into Unicode. Attribute bytes are normally translated into blanks.

The XLATE option (that can be specified using the **Set Session Parameters** function) is not supported in a Unicode session. This means that even if this option is issued, the EABs will not be translated to the PC color graphics adapter (CGA) format.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 34	
Data String	Preallocated target data string. The length should be twice the number of EBCDIC bytes required to be copied from the presentation space. When the **Set Session Parameters** (9) function with Extended Attribute Bytes (EAB) option is issued, the length of the data string must be at least four times the length of the EBCDIC field.	
Length	The length of the target data string in bytes. This length should be at least 2 in a Unicode session. If not, an error code of 2 is returned.	
PS Position	Identifies the target field. This can be the PS position of any byte within the target field. Copy always starts at the beginning of the field.	

Return Parameters

This function returns a data string, length, and a return code.

Data String:

String containing the Unicode data is returned.

Length:

Number of Unicode characters copied into string. To get the number of bytes, multiply by 2.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Copy Field to String** function was successful.
1	Your program is not connected to a host session.
2	An error was made in specifying parameters.
6	The data to be copied and the target field are not the same size. The data is truncated if the string length is smaller than the field copied.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	Unformatted host presentation space.

Notes on Using This Function

The following options are supported in a Unicode session for **Copy Field To String** and function in the same way as in SBCS:

- NOATTRB
- ATTRB
- NULLATTRB
- EAB
- NOEAB
- NOXLATE
- DISPLAY
- NODISPLAY

Copy OIA (13)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Copy OIA** function returns the current operator information area (OIA) data from the host-connected presentation space.

The OIA is located under the bottom dividing line of the screen and is used to display session status information about the connection between the workstation and the host.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 13	
Data String	Preallocated target data string	
Length	103	104
PS Position	NA	

Return Parameters

This function returns a data string and a return code.

Data String:

A 103-byte string for 16-bit and 104-byte string for 32-bit. See Format of the Returned OIA Data String for more information.

Return Code:

The following codes are defined:

Return Code	Explanation
0	OIA data is returned. The target presentation space is unlocked.
1	Your program is not connected to a host session.
2	An error was made in specifying string length. OIA data was not returned.
4	OIA data is returned. The target presentation space is busy.
5	OIA data is returned. The target presentation space is locked. (Input inhibited)
9	An internal system error was encountered. OIA data was not returned.

Notes on Using This Function

1. The OIA Group consists of the bits that show the status of the connected sessions. The group is categorized by the represented host function. (For example, Group 8 consists of the bits that show all conditions of the input inhibit in the session.) The states of each group are ordered so that the high-order bits represent the indicators of higher priority. That is, bit 7 has priority over bit 0. Therefore, if more than one state is active within a group, the state with the highest priority is the active state within that group.
2. To use this function, preallocate memory to receive the returned data string parameter. The statements required to preallocate this memory vary depending on the language in which your application is written. Refer to Memory Allocation for more information.

Format of the Returned OIA Data String

The OIA data string contains the following information:

Byte		Definition
Standard	Enhanced	
1	1	The OIA format byte. The value is 1 (PC/3270), 9 (PC400), or 5 (VT).

2-81	2-81	The OIA image in the host code points.
82-103	82-103	OIA group indicator meanings.
	104	Reserved.

PC/3270 OIA Group Indicator Meanings and Its Image

The OIA image group consists of an 80-byte ASCII character string with no attribute bytes that contains the OIA image in host code points. Figure 2 shows the hexadecimal codes found in the host presentation space, and the characters they represent. The returned data can be translated into OIA graphics characters. Refer to *Quick Beginnings* for information on the OIA indicators.

To translate the returned data into OIA graphics characters, proceed as follows:

1. Print the data returned in bytes 2 through 81 to the screen or to a printer.
2. Using the code page chart applicable to the device on which the output appears, find the hexadecimal value corresponding to each character.
3. Using Figure 2, find the OIA graphics character corresponding to each hexadecimal value found in step 2.

Note:

Group 8 (byte 0) machine, communications, and program check images are followed by a three-digit number related to the type of check.

The online and screen ownership group images are for non-SNA 3274 controller configurations. For SNA, the CD hex value is translated by CD (see Figure 2). If running on a 3174 controller or SDLC connection, the hex value X'F4' is replaced by X'B2' or X'22'. The highlight indicator is a corresponding image (in the first 80 bytes of the data string) of the "Group 5 (offset 86: Highlight group 1" byte. The highlight indicator is followed by either X'F9' (blink), X'FC' (underscore), X'D2' (reverse video), or X'80' (host default).

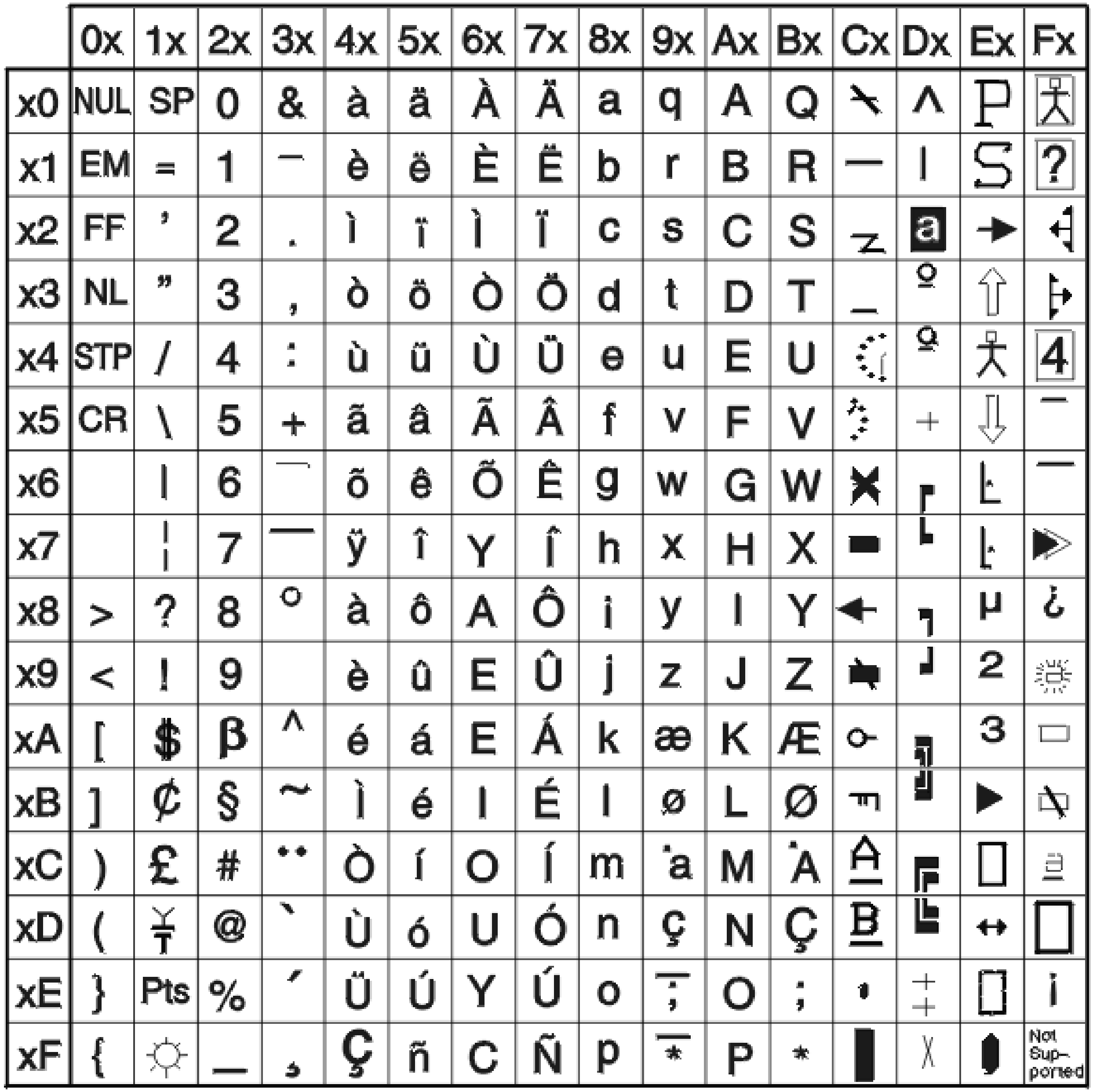
The short session ID followed by X'20' is in column 7.

All group images are represented by Main Frame Interactive (MFI) hex code points.

Note:

The OIA image data string position minus 1 position equals the OIA column.

Figure 2. Host Presentation Space Characters



	0x	1x	2x	3x	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x0	NUL	SP	0	&	à	ä	À	Ä	a	q	A	Q	⍀	^	P	웃
x1	EM	=	1	‾	è	ë	È	Ë	b	r	B	R	—	|	S	?
x2	FF	'	2	.	ì	ï	Ì	Ï	c	s	C	S	≥	a	→	⊣
x3	NL	"	3	,	ò	ö	Ò	Ö	d	t	D	T	_	♀	⇧	⊢
x4	STP	/	4	:	ù	ü	Ù	Ü	e	u	E	U	⁞	♀	웃	4
x5	CR	\	5	+	ã	â	Ã	Â	f	v	F	V	⁞	+	⇩	‾
x6		|	6	‾	õ	ê	Õ	Ê	g	w	G	W	✖	┌	┗	‾
x7		¦	7	—	ÿ	î	Y	Î	h	x	H	X	▬	╹	┗	▶
x8	>	?	8	°	à	ô	A	Ô	i	y	I	Y	←	┐	µ	¿
x9	<	!	9		è	û	E	Û	j	z	J	Z	◀	┘	2	☼
xA	[	$	β	^	é	á	E	Á	k	æ	K	Æ	⊸	]	3	▭
xB	]	¢	§	~	ì	é	I	É	l	ø	L	Ø	╥	]	▶	⊠
xC	)	£	#	¨	ò	í	O	Í	m	ȧ	M	Ȧ	A	┏	▯	⊒
xD	(	¥	@	`	Ù	ó	U	Ó	n	ç	N	Ç	B	┗	↔	▯
xE	}	Pts	%	´	Ü	Ú	Y	Ú	o	;	O	;	•	‡	▯	i
xF	{	☼	_	¸	Ç	ñ	C	Ñ	p	*	P	*	▮	╳	⬮	Not Sup-ported

- Group 1 (Offset 82): Online and Screen Ownership

Bit	Meaning
0-1	Reserved
2	SSCP-LU session owns screen
3	LU-LU session owns screen

4	Online and not owned
5	Subsystem ready
6-7	Reserved

- Group 2 (Offset 83): Character Selection

Bit	Meaning
0	Reserved
1	APL
2	Katakana (Japan only)
3	Alphanumeric
4-5	Reserved
6	Hiragana (Japan only)
7	Double-byte character

- Group 3 (Offset 84): Shift State

Bit	Meaning
0	Upper shift
1	Numeric
2	CAPS
3-7	Reserved

- Group 4 (Offset 85): PSS Group 1

Bit	Meaning
0-7	Reserved

- Group 5 (Offset 86): Highlight Group 1

Bit	Meaning
0	Operator selectable
1	Field inherit
2-7	Reserved

- Group 6 (Offset 87): Color Group 1

Bit	Meaning
0	Operator selectable
1	Field inherit
2-7	Reserved

- Group 7 (Offset 88): Insert

Bit	Meaning
0	Insert mode
1-7	Reserved

- Group 8 (Offset 89-93): Input Inhibited (5 bytes)
  - Byte 1 (Offset 89)

Bit	Meaning
0	Non-resettable machine check
1	Reserved
2	Machine check
3	Communications check
4	Program check

5-7	Reserved

- Byte 2 (Offset 90)

Bit	Meaning
0	Device busy
1	Terminal wait
2	Minus symbol
3	Minus function
4	Too much entered
5-7	Reserved

- Byte 3 (Offset 91)

Bit	Meaning
0-2	Reserved
3	Incorrect dead key combination, limited key.
4	Wrong place
5-7	Reserved

- Byte 4 (Offset 92)

Bit	Meaning
0-1	Reserved
2	System wait
3-7	Reserved

- Byte 5 (Offset 93)

Bit	Meaning
0-7	Reserved

- Group 9 (Offset 94): PSS Group 2

Bit	Meaning
0-7	Reserved

- Group 10 (Offset 95): Highlight Group 2

Bit	Meaning
0-7	Reserved

- Group 11 (Offset 96): Color Group 2

Bit	Meaning
0-7	Reserved

- Group 12 (Offset 97): Communication Error Reminder

Bit	Meaning
0-6	Communications error
1-7	Reserved

- Group 13 (Offset 98): Printer State

Bit	Meaning
0-7	Reserved

- Group 14 (Offset 99): Graphics

Bit	Meaning
0-7	Reserved

- Group 15 (Offset 100): Reserved
- Group 16 (Offset 101): Automatic Key Play/Record State

Bit	Meaning
0-7	Reserved

- Group 17 (Offset 102): Automatic Key Quit/Stop State

Bit	Meaning
0-7	Reserved

- Group 18 (Offset 103): Expanded State

Bit	Meaning
0-7	Reserved

PC400 OIA Group Indicator Meanings and Its Image

Details of the OIA group are listed in the following tables.

- Group 1 (Offset 82): Online and Screen Ownership

Bit	Meaning	Beginning Position of Data String
0-2	Reserved	
3	System available	1
4	Reserved	
5	Subsystem ready	
6-7	Reserved	

- Group 2 (Offset 83): Character Selection

Bit	Meaning	Beginning Position of Data String
0-1	Reserved	
2	Katakana (Japan only)	
3	Alphanumeric	
4-5	Reserved	
6	Hiragana (Japan only)	
7	Double-byte character	

- Group 3 (Offset 84): Shift State

Bit	Meaning	Beginning Position of Data String
0	Reserved	
1	Keyboard shift	39
2	CAPS	
3-6	Reserved	
7	Double-byte character input available	

- Group 4 (Offset 85): PSS Group 1

Bit	Meaning	Beginning Position of Data String

0-7	Reserved	

- Group 5 (Offset 86): Highlight Group 1

Bit	Meaning	Beginning Position of Data String
0-7	Reserved	

- Group 6 (Offset 87): Color Group 1

Bit	Meaning	Beginning Position of Data String
0-7	Reserved	

- Group 7 (Offset 88): Insert

Bit	Meaning	Beginning Position of Data String
0	Insert mode	68
1-7	Reserved	

- Group 8 (Offset 89-93): Input Inhibited (5 bytes)
  - Byte 1 (Offset 89)

Bit	Meaning	Beginning Position of Data String
0-7	Reserved	

  - Byte 2 (Offset 90)

Bit	Meaning	Beginning Position of Data String
0-7	Reserved	

  - Byte 3 (Offset 91)

Bit	Meaning	Beginning Position of Data String
0-4	Reserved	
5	Operator input error	64
6-7	Reserved	

  - Byte 4 (Offset 92)

Bit	Meaning	Beginning Position of Data String
0-1	Reserved	
2	System wait	64
3-7	Reserved	

  - Byte 5 (Offset 93)

Bit	Meaning	Beginning Position of Data String
0-7	Reserved	

- Group 9 (Offset 94): PSS Group 2

Bit	Meaning	Beginning Position of Data String
0-7	Reserved	

- Group 10 (Offset 95): Highlight Group 2

Bit	Meaning	Beginning Position of Data String
0-7	Reserved	

- Group 11 (Offset 96): Color Group 2

Bit	Meaning	Beginning Position of Data String
0-7	Reserved	

- Group 12 (Offset 97): Communication Error Reminder

Bit	Meaning	Beginning Position of Data String
0	Communications Error	
1-5	Reserved	
7	Message wait	3

- Group 13 (Offset 98): Printer State

Bit	Meaning	Beginning Position of Data String
0-7	Reserved	

- Group 14 (Offset 99): Graphics

Bit	Meaning	Beginning Position of Data String
0-7	Reserved	

- Group 15 (Offset 100): Reserved
- Group 16 (Offset 101): Automatic Key Play/Record State

Bit	Meaning	Beginning Position of Data String
0-7	Reserved	

- Group 17 (Offset 102): Automatic Key Quit/Stop State

Bit	Meaning	Beginning Position of Data String
0-7	Reserved	

- Group 18 (Offset 103): Expanded State

Bit	Meaning	Beginning Position of Data String
0-7	Reserved	

VT Host OIA Group Indicator Meanings and Its Image

Details of the VT Host OIA group are listed in the following tables.

- Group 1 (Offset 82): Online and Screen Ownership

Bit	Meaning
5	Subsystem ready

- Group 2 (Offset 83): Character Selection

Bit	Meaning
0	Upper shift
2	CAPS

- Group 7 (Offset 88): Insert

Bit	Meaning
0	Insert mode

Some columns on the OIA line display different messages for VT than those messages displayed for 3270/5250. See the following table for specific details.

Column	Symbol
1-7	VT220 7
	VT220 8
	VT100
	VT52
	VTANSI
9 - 12	LOCK
61 - 64	HOLD

Copy Presentation Space (5)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Copy Presentation Space** function copies the contents of the host-connected presentation space into a data string that you define in your EHLLAPI application program.

The **Copy Presentation Space** function translates the characters in the host source presentation space into ASCII. Attribute bytes and other characters not represented in ASCII normally are translated into blanks. If you do not want the attribute bytes translated into blanks, you can override this translation with the ATTRB option under the **Set Session Parameters** (9) function.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 5	
Data String	Preallocated target string the size of your host presentation space. This can vary depending on how your host presentation space is configured. When the **Set Session Parameters** (9) function with the EAB option is issued, the length of the data string must be at least twice the length of the presentation space. **DBCS Only:** When the EAD option is specified, the length of the data string must be at least three times the length of the presentation space. When both the EAB and EAD options are specified, the length of the data string must be at least four times the length of the presentation space.	
Length	NA (the length of the host presentation space is implied).	
PS Position	NA.	

Return Parameters

This function returns a data string, length, and a return code.

Data String:

Contents of the connected host presentation space.

Length:

Length of the data copied.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The host presentation space contents were copied to the application program. The target presentation space was active, and the keyboard was unlocked.
1	Your program is not connected to a host session.
4	The host presentation space contents were copied. The connected host presentation space was waiting for host response.
5	The host presentation space was copied. The keyboard was locked.
9	A system error was encountered.

Notes on Using This Function

1. An EAB can be returned when the **Set Session Parameters** (9) function EAB option is used. EAB is related to each character in the presentation space and is returned preceding each character.
2. **DBCS Only:** The **Set Session Parameters** (9) function EAD option is used with this function to return a 2-byte EAD. If the EAD option is specified instead of the EAB option, EAD is returned preceding each character. If both the EAB and EAD options are specified, EAD is returned preceding the EAB.

   If the start position of the copy is at the second byte in the double-byte character, or the end position is at the first byte in the double-byte character, the bytes are translated into blanks.

3. The **Copy Presentation Space** function is affected by the following session options:
   - ATTRB/NOATTRB/NULLATTRB
   - EAB/NOEAB
   - XLATE/NOXLATE
   - BLANK/NOBLANK
   - DISPLAY/NODISPLAY
   - EAD/NOEAD (for DBCS only)
   - NOSO/SPACESO/SO (for DBCS only)
   - EXTEND_PS/NOEXTEND_PS

Refer to items 5; 13, 14, 15 and 17; and 20 and 21 for more information.

If the target data string provided is not long enough to hold the requested data, unpredictable results can occur.

As previously stated, the return of attributes by the various **Copy** (5, 8, and 34) functions is affected by the **Set Session Parameters** (9) function. The involved set session parameters have the following effect:

Set Session Parameter

Effect on the COPY Function

NOEAB and NOEAD

Attributes are not returned. Only text is copied from the presentation space to the user buffer.

EAB and NOXLATE

Attributes are returned as defined in the following tables.

EAB and XLATE

The colors used for the presentation space display are returned. Colors can be remapped; so the attribute colors are not the ones returned by the **Copy** functions when XLATE and EAB are on at the same time.

EAD

Double-byte character set attributes are returned as shown in the following tables.

NOSO/SPACESO/SO

When NOSO is specified, it works as SPACESO. The size of the presentation space is not changed.

The returned character attributes are defined in the following tables. The attribute bit positions are in IBM format with bit 0 the left most bit in the byte.

3270 character attributes are returned from the host to the emulator. The following table applies when EAB and NOXLATE are set.

Bit Position	Meaning
0-1	Character highlighting 00 = Normal 01 = Blink 10 = Reverse video 11 = Underline
2-4	Character color (Color remap can override this color definition.) 000 = Default 001 = Blue 010 = Red 011 = Pink 100 = Green 101 = Turquoise 110 = Yellow 111 = White
5-6	Character attribute 00 = Default value 11 = Double-byte character
7	Reserved

5250 character attributes are returned from the host to the emulator. The following table applies when EAB and NOXLATE are set.

Bit Position	Meaning
0	Reverse image 0 = Normal image 1 = Reverse image
1	Underline 0 = No underline 1 = Underline
2	Blink 0 = Not blink 1 = Blink
3	Separator of columns 0 = No separator 1 = Separator
4-7	Reserved

The following table shows Personal Communications character color attributes. The following table applies when EAB and XLATE are set.

Bit Position	Meaning
0-3	Background character colors 0000 = Black 0001 = Blue 0010 = Green 0011 = Cyan 0100 = Red 0101 = Magenta 0110 = Brown (3270), Yellow (5250) 0111 = White
4-7	Foreground character colors 0000 = Black 0001 = Blue 0010 = Green 0011 = Cyan 0100 = Red 0101 = Magenta 0110 = Brown (3270), Yellow (5250)

	0111 = White 1000 = Gray 1001 = Light blue 1010 = Light green 1011 = Light cyan 1100 = Light red 1101 = Light magenta 1110 = Yellow 1111 = White (high intensity)

- Double-byte character set attributes (for DBCS only)
  - The first byte

Bit Position	Character Position	Field Attribute Position
0	Double-byte character	Reserved
1	The first byte of the double-byte character	Reserved
2	SO	Reserved
3-4	SI (Bit position 3)	5250 DBCS related field ▪ When the value of bit position 7 is 0: 00 = Default 01 = DBCS only 10 = Either DBCS or SBCS 11 = Mixture of DBCS and SBCS ▪ When the value of bit position 7 is 1: 00 = Reserved 01 = DBCS only without SO/SI 10 = Reserved 11 = Reserved
5	Reserved	SO/SI enabled (3270 only)
6	Reserved	Character attributes exist (3270 only)
7	Reserved	5250 DBCS related extended field 0 = Basic double-byte field 1 = Extended double-byte field

  - The second byte

Bit Position	Character Position	Field Attribute Position
0	Reserved	Left grid line (3270 only)
1	Reserved	Upper grid line (3270 only)
2	Reserved	Right grid line (3270 only)
3	Reserved	Under grid line (3270 only)
4	Left grid line	Left grid line

5	Upper grid line	Upper grid line
6-7	Reserved	Reserved

For a PS/2 monochrome display, the characters in the application (workstation) session appear as various shades of gray. This is required to give users their remapped colors in the EHLLAPI application session so they can get what they see in their host application presentation spaces.

If you want to copy only a portion of the host presentation space, use the **Copy Presentation Space to String** (8) function.

To use this function, preallocate memory to receive the returned data string parameter. The statements required to preallocate this memory vary depending on the language in which your application is written. Refer to Memory Allocation for more information.

Note:

5250 emulation supports a presentation space of 24 rows by 80 columns. In some instances, Communication Manager 5250 emulation displays a 25th row. This occurs when either an error message from the host is displayed or when the operator selects the SysReq key. Personal Communications displays 25th row information on row 24, or on the status bar. For information to be displayed on the status bar, the status bar must be configured. Refer to *Quick Beginnings* for information on configuring the status bar. By the **EXTEND_PS** option, an EHLLAPI application can use the same interface with Communication Manager EHLLAPI and valid presentation space is extended when this condition occurs.

1390/1399 Code Page Support

Unicode functionality is supported only on 3270 and 5250 sessions.

In a Unicode session, the characters in the host source presentation space are translated into Unicode. Attribute bytes are normally translated into blanks.

The XLATE option (that can be specified using the **Set Session Parameters** (9) function) is not supported in a Unicode session. This means that even if this option is issued, the EABs will not be translated to the PC color graphics adapter (CGA) format.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 5	
Data String	Preallocated target Unicode string. When the **Set Session Parameters** (9) function with Extended Attribute Bytes (EAB) option is issued, the length of the data string must be twice the size of the presentation space.	
Length	NA (the length of the host presentation space is implied).	
PS Position	NA	

Return Parameters

This function returns a data string and a return code.

Data String:

String containing the Unicode representation of the contents of presentation space is returned

Return Code:

The following codes are defined:

Return Code	Explanation
0	The host presentation space contents were copied to the application program. The target presentation space was active, and the keyboard was unlocked.
1	Your program is not connected to a host session.
4	The host presentation space contents were copied. The connected host presentation space was waiting for host response.
5	The host presentation space was copied. The keyboard was locked.
9	A system error was encountered.

Notes on Using This Function

The following options are supported in a Unicode session for **Copy Presentation Space (5)** and function in the same way as in DBCS:

- NOATTRB
- ATTRB
- NULLATTRB
- EAB
- NOEAB
- NOXLATE
- DISPLAY
- NODISPLAY
- BLANK
- NOBLANK

1137 Code Page Support

Unicode functionality is supported only on 5250 sessions.

In a Unicode session, the characters in the host source presentation space are translated into Unicode. Attribute bytes are normally translated into blanks.

The XLATE option (that can be specified using the **Set Session Parameters** (9) function) is not supported in a Unicode session. This means that even if this option is issued, the EABs will not be translated to the PC color graphics adapter (CGA) format.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 5	
Data String	Preallocated target Unicode data string. The length (in bytes) should be twice the size (in bytes) of the presentation space. When the **Set Session Parameters** (9) function with Extended Attribute Bytes (EAB) option is issued, the length of the data string must be at least four times the size of the presentation space.	

Length	NA (the length of the host presentation space is implied).
PS Position	NA

Return Parameters

This function returns a data string and a return code.

Data String:

String containing the Unicode representation of the contents of presentation space is returned

Return Code:

The following codes are defined:

Return Code	Explanation
0	The host presentation space contents were copied to the application program. The target presentation space was active, and the keyboard was unlocked.
1	Your program is not connected to a host session.
4	The host presentation space contents were copied. The connected host presentation space was waiting for host response.
5	The host presentation space was copied. The keyboard was locked.
9	A system error was encountered.

Notes on Using This Function

The following options are supported in a Unicode session for **Copy Presentation Space (5)** and function in the same way as in SBCS:

- NOATTRB
- ATTRB
- NULLATTRB
- EAB
- NOEAB
- NOXLATE
- DISPLAY
- NODISPLAY
- BLANK
- NOBLANK

Copy Presentation Space to String (8)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Copy Presentation Space to String** function is used to copy all or part of the host-connected presentation space into a data string that you define in your EHLLAPI application program.

The input PS position is the offset into the host presentation space. This offset is based on a layout in which the upper-left corner (row 1/column 1) is location 1 and the bottom-right corner is 3564, which is the maximum screen size for the host presentation space. The value of `PS Position + (Length - 1)` cannot exceed the configured size of your host presentation space.

The **Copy Presentation Space to String** function translates the characters in the host source presentation space into ASCII. Attribute bytes and other characters not represented in ASCII normally are translated into blanks. If you do not want the attribute bytes translated into blanks, you can override this translation with the ATTRB option under the **Set Session Parameters** (9) function.

Prerequisite Calls

Connect Presentation Space (1).

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 8	
Data String	Preallocated target string the size of your host presentation space. When the **Set Session Parameters** (9) function with the EAB option is issued, the length of the data string must be at least twice the length of the presentation space. **DBCS Only:** When the EAD option is specified, the length of the data string must be at least three times the length of the presentation space. When both the EAB and EAD options are specified, the length of the data string must be at least four times the length of the presentation space.	
Length	Length of the target data string.	
PS Position	Position within the host presentation space of the first byte in your target data string.	

Return Parameters

This function returns a data string and a return code.

Data String:

Contents of the host presentation space.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The host presentation space contents were copied to the application program. The target presentation space was active, and the keyboard was unlocked.
1	Your program is not connected to a host session.
2	An error was made in specifying string length, or the sum of (Length - 1) + PS position is greater than the size of the connected host presentation space.
4	The host presentation space contents were copied. The host presentation space was waiting for host response.
5	The host presentation space was copied. The keyboard was locked.
7	The host presentation space position is not valid.
9	A system error was encountered.

Notes on Using This Function

1. An EAB can be returned when the **Set Session Parameters** (9) function EAB option is used. EAB is related to each character in the presentation space and is returned following each character.

2. **DBCS Only:** The **Set Session Parameters** (9) function EAD option is used with this function to return a 2-byte EAD. If the EAD option is specified instead of the EAB option, EAD is returned preceding each character. If both the EAB and EAD options are specified, EAD is returned following the EAB.

   If the start position of the copy is at the second byte in the double-byte character, or the end position is at the first byte in the double-byte character, the bytes are translated into blanks. If the EAD option is set to on, three bytes are returned for each character. If both the EAB and EAD options are set to on, four bytes are returned for each character.

3. The **Copy Presentation Space to String** function is affected by the following options:
   - ATTRB/NOATTRB/NULLATTRB
   - EAB/NOEAB
   - XLATE/NOXLATE
   - BLANK/NOBLANK
   - DISPLAY/NODISPLAY
   - EAD/NOEAD (for DBCS only)
   - NOSO/SPACESO/SO (for DBCS only)
   - EXTEND_PS/NOEXTEND_PS

   Refer to items 5; 13 and 14; 15; 17; and 20 and 21

   If the target data string provided is not large enough to hold the requested number of bytes, the copy ends successfully (RC=0, 4, or 5) when the end of the target data string is reached.

   As previously stated, the return of attributes by the various **Copy** (5, 8, and 34) functions is affected by the **Set Session Parameters** (9) function. The involved set session parameters have the following effect:

   Set Session Parameter

   Effect on the Copy Function

   NOEAB and NOEAD

   Attributes are not returned. Only text is copied from the presentation space to the user buffer.

   EAB and NOXLATE

   Attributes are returned as defined in the following tables.

   EAB and XLATE

   The colors used for the presentation space display are returned. Colors can be remapped, so the attribute colors are not the ones returned by the **Copy** functions when XLATE and EAB are on at the same time.

   EAD

   Double-byte character set attributes are returned as shown in the following tables.

   The returned character attributes are defined in the following tables. The attribute bit positions are in IBM format with bit 0 the left most bit in the byte.

   - 3270 character attributes are returned from the host to the emulator. The following table applies when EAB and NOXLATE are set.

Bit Position	Meaning
0-1	Character highlighting 00 = Normal 01 = Blink 10 = Reverse video 11 = Underline
2-4	Character color (Color remap can override this color definition.) 000 = Default 001 = Blue 010 = Red 011 = Pink 100 = Green 101 = Turquoise 110 = Yellow 111 = White
5-7	Reserved

- 5250 character attributes are returned from the host to the emulator. The following table applies when EAB and NOXLATE are set.

Bit Position	Meaning
0	Reverse image 0 = Normal image 1 = Reverse image
1	Underline 0 = No underline 1 = Underline
2	Blink 0 = Not blink 1 = Blink
3	Separator of columns 0 = No separator 1 = Separator
4-7	Reserved

- VT character attributes are returned from the host to the emulator. The following table applies when EAB and NOXLATE are set.

Bit Position	Meaning
0-3	Reserved
4	Bold 1 = On 0 = Off
5	Underscore 1 = On 1 = Off
6	Blink 1 = On 0 = Off
7	Reverse 0 = On 1 = Off

- The following table shows Personal Communications character color attributes. The following table applies when EAB and XLATE are set.

Bit Position	Meaning
0-3	Background character colors 0000 = Black 0001 = Blue 0010 = Green 0011 = Cyan 0100 = Red 0101 = Magenta 0110 = Brown (3270), Yellow (5250) 0111 = White
4-7	Foreground character colors 0000 = Black 0001 = Blue 0010 = Green 0011 = Cyan 0100 = Red 0101 = Magenta 0110 = Brown (3270), Yellow (5250)

	0111 = White 1000 = Gray 1001 = Light blue 1010 = Light green 1011 = Light cyan 1100 = Light red 1101 = Light magenta 1110 = Yellow 1111 = White (high intensity)

- Double-byte character set attributes
  - The first byte

Bit Position	Character Position	Field Attribute Position
0	Double-byte character	Reserved
1	The first byte of the double-byte character	Reserved
2	SO	Reserved
3-4	SI (Bit position 3)	5250 DBCS related field When the value of bit position 7 is 0: 00 = Default 01 = DBCS only 10 = Either DBCS or SBCS 11 = Mixture of DBCS and SBCS When the value of bit position 7 is 1: 00 = Reserved 01 = DBCS only without SO/SI 10 = Reserved 11 = Reserved
5	Reserved	SO/SI enable (3270 only)
6	Reserved	Character Attributes exist (3270 only)
7	Reserved	5250 DBCS related extended field 0 = Basic double-byte field 1 = Extended double-byte field

  - The second byte

Bit Position	Character Position	Field Attribute Position
0	Reserved	Left grid line (3270 only)
1	Reserved	Upper grid line (3270 only)
2	Reserved	Right grid line (3270 only)
3	Reserved	Under grid line (3270 only)

4	Left grid line	Left grid line
5	Upper grid line	Upper grid line
6-7	Reserved	Reserved

For a PS/2 monochrome display, the characters in the application (workstation) session appear as various shades of gray. This is required to give users their remapped colors in the EHLLAPI application session so they can get what they see in their host application presentation spaces.

4. To use this function, preallocate memory to receive the returned data string parameter. The statements required to preallocate this memory vary depending on the language in which your application is written. Refer to Memory Allocation for more information.

Note:

5250 emulation supports a presentation space of 24 rows by 80 columns. In some instances, Communication Manager 5250 emulation displays a 25th row. This occurs when either an error message from the host is displayed or when the operator selects the SysReq key. Personal Communications displays 25th row information on row 24, or on the status bar. For information to be displayed on the status bar, the status bar must be configured. Refer to *Quick Beginnings* for information on configuring the status bar. By the **EXTEND_PS** option, an EHLLAPI application can use the same interface with Communication Manager EHLLAPI and valid presentation space is extended when this condition occurs.

1390/1399 Code Page Support

Unicode functionality is supported only on 3270 and 5250 sessions.

In a Unicode session, the characters in the host source presentation space are translated into Unicode. Attribute bytes are normally translated into blanks.

The XLATE option (that can be specified using the **Set Session Parameters** (9) function) is not supported in a Unicode session. This means that even if this option is issued, the EABs will not be translated to the PC color graphics adapter (CGA) format.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 8	
Data String	Preallocated target Unicode string. When the **Set Session Parameters** (9) function with Extended Attribute Bytes (EAB) option is issued, the length of the data string must be at least twice the length of the presentation space.	
Length	The length of the target Unicode string in Unicode characters.	
PS Position	Position within the host presentation space of the first byte in your target data string.	

Return Parameters

This function returns a data string and a return code.

Data String:

String containing the Unicode data is returned

Return Code:

The following codes are defined:

Return Code	Explanation
0	The host presentation space contents were copied to the application program. The target presentation space was active, and the keyboard was unlocked.
1	Your program is not connected to a host session.
2	An error was made in specifying string length, or the sum of (Length - 1) + PS position is greater than the size of the connected host presentation space.
4	The host presentation space contents were copied. The host presentation space was waiting for host response.
5	The host presentation space was copied. The keyboard was locked.
7	The host presentation space position is not valid.
9	A system error was encountered.

Notes on Using This Function

The following options are supported in a Unicode session for **Copy Presentation Space to String** and function in the same way as in DBCS:

- NOATTRB
- ATTRB
- NULLATTRB
- EAB
- NOEAB
- NOXLATE
- DISPLAY
- NODISPLAY
- BLANK
- NOBLANK

1137 Code Page Support

Unicode functionality is supported only on 5250 sessions.

In a Unicode session, the characters in the host source presentation space are translated into Unicode. Attribute bytes are normally translated into blanks.

The XLATE option (that can be specified using the **Set Session Parameters** (9) function) is not supported in a Unicode session. This means that even if this option is issued, the EABs will not be translated to the PC color graphics adapter (CGA) format.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 8	
Data String	Preallocated target data string. The length should be at least twice the number of EBCDIC bytes required to be copied from the presentation space.	

	When the **Set Session Parameters** (9) function with Extended Attribute Bytes (EAB) option is issued, the length of the data string must be at least four times the length of the EBCDIC string that is to be copied from the presentation space.
Length	The length of the target Unicode string in bytes. This length should be at least 2 in a Unicode session. If not, an error code of 2 is returned.
PS Position	Position within the host presentation space of the first byte in your target data string.

Return Parameters

This function returns a data string and a return code.

Data String:

Contents of the host presentation space.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The host presentation space contents were copied to the application program. The target presentation space was active, and the keyboard was unlocked.
1	Your program is not connected to a host session.
2	An error was made in specifying string length, or the sum of (Length - 1) + PS position is greater than the size of the connected host presentation space.
4	The host presentation space contents were copied. The host presentation space was waiting for host response.
5	The host presentation space was copied. The keyboard was locked.
7	The host presentation space position is not valid.
9	A system error was encountered.

Notes on Using This Function

The following options are supported in a Unicode session for **Copy Presentation Space to String** and function in the same way as in SBCS:

- NOATTRB
- ATTRB
- NULLATTRB
- EAB
- NOEAB
- NOXLATE
- DISPLAY
- NODISPLAY
- BLANK
- NOBLANK

Copy String to Field (33)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Copy String to Field** function transfers a string of characters into a specified field in the host-connected presentation space. This function can be used only in a *field-formatted* host presentation space.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 33	
Data String	String containing the data to be transferred to a target field in the host presentation space.	
Length	Length, in number of bytes, of the source data string. Overridden if in EOT mode.	
PS Position	Identifies the target field. This can be the PS position of any byte within the target field. Copy always starts at the beginning of the field.	

Return Parameters

Return Code	Explanation
0	The **Copy String to Field** function was successful.
1	Your program is not connected to a host session.
2	Parameter error or zero length for copy.
5	The target field was protected or inhibited, or incorrect data was sent to the target field (such as a field attribute).
6	Copy was completed, but data is truncated.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	Unformatted host presentation space.

Notes on Using This Function

1. The **Copy String to Field** function is affected by the following options:
   - STRLEN/STREOT
   - EOT
   - EAB/NOEAB
   - XLATE/NOXLATE
   - PUTEAB/NOPUTEAB

   Refer to items 1 and 2; 13 and 14; 18; and 20 and 21 for more information.

2. The string to be transferred is specified with the calling data string parameter. The string ends when one of these three conditions is encountered:
   - When an end-of-text (EOT) delimiter is encountered in the string if EOT mode was selected using the **Set Session Parameters** (9) function. (See Set Session Parameters (9)).
   - When the number specified in the length is reached if not in EOT mode.
   - When an end-of-field is encountered in the field.

   Note:

   If the field at the end of the host presentation space wraps, wrapping occurs when the end of the presentation space is reached.

3. The keyboard mnemonics (see **Send Key** (3) function) cannot be sent using the **Copy String to Field** function.
4. The first byte of the data to be transferred is always placed at the beginning of the field that contains the specified PS position.
5. **DBCS Only:** Double-byte characters can be included as a part of the string.

   Note:

   PC400 does not add SO and SI to the string. When you write the strings, including double-byte characters at the DBCS mixed field, generate SO and SI and create the area where double-byte characters are written by using the **Send Key** (3) function in advance.

   If both single-byte and double-byte characters exist in a string, the data might be truncated because the data length in EBCDIC is longer than in JISCII. In this case, only the first byte or the second byte of the double-byte character is not written.

   If the last character in the original string is the first byte of the double-byte character, the character is not written and not counted in the length.

   A control character is converted from single-byte character to double-byte character, or from double-byte character to single-byte character depending on the field condition. A pair of NULL+Control Character between SO and SI is treated as a double-byte control character. For example, the following strings are copied into the single-byte character field or the double-byte character field:

String	Meanings	Single-byte character field	Double-byte character field
X'000C'	(NULL)(FF) X'00'X'0C'	(SB NULL)(SB FF) X'00'X'0C'	(DB NULL)(DB FF) X'0000'X'000C'
X'0E000C0F'	(SO)(DB FF)(SI) X'0E'X'000C'X'0F'	-S error	(DB FF) X'000C'

Note:

SB means single-byte characters and DB means double-byte characters.

Note:

5250 emulation supports a presentation space of 24 rows by 80 columns. In some instances, Communication Manager 5250 emulation displays a 25th row. This occurs when either an error message from the host is displayed or when the operator selects the SysReq key. Personal Communications displays 25th row information on row 24, or on the status bar. For information to be displayed on the status bar, the status bar must be configured. Refer to *Quick Beginnings* for information on configuring the status bar. By the **EXTEND_PS** option, an EHLLAPI application can use the same interface with Communication Manager EHLLAPI and valid presentation space is extended when this condition occurs.

1390/1399 Code Page Support

Unicode functionality is supported only on 3270 and 5250 sessions.

STREOT option is not supported in a Unicode session. Please refer to Set Session Parameters (9) for details.

The XLATE option (that can be specified using the **Set Session Parameters** (9) function) is not supported in a Unicode session. This means that even if this option is issued, the EABs will not be translated to the PC color graphics adapter (CGA) format.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 33	
Data String	String containing the Unicode data to be transferred to a target field in the host presentation space.	
Length	Length, in number of Unicode characters, of the source Unicode string. **Note:** The EOT mode is not supported in a Unicode session; therefore, length should be specified for proper functioning of this function in a Unicode session.	
PS Position	Identifies the target field. This can be the PS position of any byte within the target field. Copy always starts at the beginning of the field.	

Return Parameters

Return Code	Explanation
0	The **Copy String to Field** function was successful.
1	Your program is not connected to a host session.
2	Parameter error or zero length for copy.
5	The target field was protected or inhibited, or incorrect data was sent to the target field (such as a field attribute).
6	Copy was completed, but data is truncated.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	Unformatted host presentation space.

Notes on Using This Function

The following options are supported in a Unicode session for **Copy String to Field** and function in the same way as in DBCS:

- STRLEN
- EAB
- NOEAB
- NOXLATE
- PUTEAB
- NOPUTEAB

1137 Code Page Support

Unicode functionality is supported only on 5250 sessions.

STREOT option is not supported in a Unicode session. Please refer to Set Session Parameters (9) for details.

The XLATE option (that can be specified using the **Set Session Parameters** (9) function) is not supported in a Unicode session. This means that even if this option is issued, the EABs will not be translated to the PC color graphics adapter (CGA) format.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 33	
Data String	String containing the Unicode data to be transferred to a target field in the host presentation space.	
Length	Length, in number of bytes, of the source Unicode string. The length should be at least 2 bytes. If not, an error code of 2 is returned. **Note:** The EOT mode is not supported in a Unicode session; therefore, length should be specified for proper functioning of this function in a Unicode session.	
PS Position	Identifies the target field. This can be the PS position of any byte within the target field. Copy always starts at the beginning of the field.	

Return Parameters

Return Code	Explanation
0	The **Copy String to Field** function was successful.
1	Your program is not connected to a host session.
2	Parameter error or zero length for copy.
5	The target field was protected or inhibited, or incorrect data was sent to the target field (such as a field attribute).
6	Copy was completed, but data is truncated.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	Unformatted host presentation space.

Notes on Using This Function

The following options are supported in a Unicode session for **Copy String to Field** and function in the same way as in SBCS:

- STRLEN
- EAB
- NOEAB
- NOXLATE
- PUTEAB
- NOPUTEAB

Copy String to Presentation Space (15)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Copy String to Presentation Space** function copies an ASCII data string directly into the host presentation space at the location specified by the PS position calling parameter.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 15.	
Data String	String of ASCII data to be copied into the host presentation space.	
Length	Length, in number of bytes, of the source data string. Overridden if in EOT mode.	
PS Position	Position in the host presentation space to begin the copy, a value between 1 and the configured size of your host presentation space.	

Return Parameters

Return Code	Explanation
0	The **Copy String to Presentation Space** function was successful.
1	Your program is not connected to a host session.
2	Parameter error or zero length for copy.
5	The target presentation space is protected or inhibited, or incorrect data was sent to the target presentation space (such as a field attribute byte).
6	The copy was completed, but the data was truncated.
7	The host presentation space position is not valid.
9	A system error was encountered.

Notes on Using This Function

1. The **Copy String to Presentation Space** function is affected by the following options:
   - STRLEN/STREOT
   - EOT
   - EAB/NOEAB
   - XLATE/NOXLATE
   - PUTEAB/NOPUTEAB
   - EAD/NOEAD (for DBCS only)
   - NOSO/SPACESO/SO (for DBCS only)
   - EXTEND_PS/NOEXTEND_PS

   Refer to items 1 and 2; 13 and 14; 18; and 20 and 21 for more information.

2. The keyboard mnemonics (see **Send Key** (3) function) cannot be sent using the **Copy String to Presentation Space** function.
3. The string ends when an end-of-text (EOT) delimiter is encountered in the string if EOT mode was selected using the **Set Session Parameters** (9) function. (See Set Session Parameters (9)).
4. Although the **Send Key** (3) function accomplishes the same purpose, this function responds with the prompt and enters a command more quickly. Because the **Send Key** (3) function emulates the terminal operator typing the data from the keyboard, its process speed is slow for an application operating with a lot of data. This function provides a faster input path to the host.
5. The original data (the copied string) cannot exceed the size of the presentation space.
6. **DBCS Only:** Double-byte characters can be included as a part of the string.

   Note:

   > PC400 does not add SO and SI to the string. When you write the strings, including double-byte characters at the DBCS mixed field, generate SO and SI and create the area where double-byte characters are written by using the **Send Key** (3) function in advance.

   If both single-byte and double-byte characters exist in a string, the data might be truncated because the data length in EBCDIC is longer than in JISCII. If only the first byte or the second byte of the double-byte character must be written into the string, a blank is written.

If the last character in the original string is the first byte of the double-byte character, the character is not written and not counted in the length.

If the character to be written into the last character of the target presentation space is SO/SI or the first byte of the double-byte character, the character is not written and truncated, and not counted in the length.

A control character is converted from single-byte character to double-byte character, or from double-byte character to single-byte character depending on the field condition. A pair of NULL+Control Character between SO and SI is treated as a double-byte control character. For example, the following strings are copied into the single-byte character field or the double-byte character field:

String	Meanings	Single-byte character field	Double-byte character field
X'000C'	(NULL)(FF) X'00'X'0C'	(SB NULL)(SB FF) X'00'X'0C'	(DB NULL)(DB FF) X'0000'X'000C'
X'0E000C0F'	(SO)(DB FF)(SI) X'0E'X'000C'X'0F'	-S error	(DB FF) X'000C'

Note:

SB means single-byte characters and DB means double-byte characters.

Note:

5250 emulation supports a presentation space of 24 rows by 80 columns. In some instances, Communication Manager 5250 emulation displays a 25th row. This occurs when either an error message from the host is displayed or when the operator selects the SysReq key. Personal Communications always displays the same information on the 24th row. By the **EXTEND_PS** option, an EHLLAPI application can use the same interface with Communication Manager EHLLAPI and valid presentation space is extended when this condition occurs.

7. This function call may cause a cursor movement to an unexpected position with some host applications. A SendKey function may be a better choice for filling a field than this function.

Note:

This only occurs with VT sessions or connections to an ASCII host.

1390/1399 Code Page Support

Unicode functionality is supported only on 3270 and 5250 sessions.

STREOT option is not supported in a Unicode session. Please refer to Set Session Parameters (9) for details.

The XLATE option (that can be specified using the **Set Session Parameters** (9) function) is not supported in a Unicode session. This means that even if this option is issued, the EABs will not be translated to the PC color graphics adapter (CGA) format.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 15	
Data String	String containing the Unicode data to be transferred into the host presentation space.	
Length	Length, in number of Unicode characters, of the source Unicode string. **Note:** The EOT mode is not supported in a Unicode session; therefore, length should be specified for proper functioning of this function in a Unicode session.	
PS Position	Position in the host presentation space to begin the copy, a value between 1 and the configured size of your host presentation space.	

Return Parameters

Return Code	Explanation
0	The **Copy String to Presentation Space** function was successful.
1	Your program is not connected to a host session.
2	Parameter error or zero length for copy.
5	The target presentation space is protected or inhibited, or incorrect data was sent to the target presentation space (such as a field attribute byte).
6	The copy was completed, but the data was truncated.
7	The host presentation space position is not valid.
9	A system error was encountered.

Notes on Using This Function

The following options are supported in a Unicode session for **Copy String to Presentation Space** and function in the same way as in DBCS:

- STRLEN
- EAB
- NOEAB
- NOXLATE
- PUTEAB
- NOPUTEAB

1137 Code Page Support

Unicode functionality is supported only on 5250 sessions.

STREOT option is not supported in a Unicode session. Please refer to Set Session Parameters (9) for details.

The XLATE option (that can be specified using the **Set Session Parameters** (9) function) is not supported in a Unicode session. This means that even if this option is issued, the EABs will not be translated to the PC color graphics adapter (CGA) format.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 15	
Data String	String containing the Unicode data to be transferred into the host presentation space.	
Length	Length, in number of Unicode characters, of the source Unicode string. The length should be at least 2 bytes. If not, an error code of 2 is retuned. **Note:** The EOT mode is not supported in a Unicode session; therefore, length should be specified for proper functioning of this function in a Unicode session.	
PS Position	Position in the host presentation space to begin the copy, a value between 1 and the configured size of your host presentation space.	

Return Parameters

Return Code	Explanation
0	The **Copy String to Presentation Space** function was successful.
1	Your program is not connected to a host session.
2	Parameter error or zero length for copy.
5	The target presentation space is protected or inhibited, or incorrect data was sent to the target presentation space (such as a field attribute byte).
6	The copy was completed, but the data was truncated.
7	The host presentation space position is not valid.
9	A system error was encountered.

Notes on Using This Function

The following options are supported in a Unicode session for **Copy String to Presentation Space** and function in the same way as in SBCS:

- STRLEN
- EAB
- NOEAB
- NOXLATE
- PUTEAB
- NOPUTEAB

Disconnect from Structured Fields (121)

3270	*5250*	*VT*
Yes	No	No

The **Disconnect from Structured Fields** function drops the connection between the emulation program and the EHLLAPI application. The EHLLAPI application must disconnect from the emulation program before exiting from the system. The EHLLAPI application should issue this function request if a previous **Connect for Structured Fields** was issued.

The **Reset System (21)** function will also disconnect any outstanding SF connections.

Prerequisite Calls

Connect for Structured Fields (120)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 121	
Data String	See the following table	
Length	Must be 3	Must be 8
PS Position	NA	

Data String Contents

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID).
	2-4	Reserved.
2-3	5-6	Destination/origin unique ID returned by the Connect for structured field (120) functions.
	7-8	Reserved.

Return Parameters

Return Code	Explanation
0	The **Disconnect from Structured Fields** function was successful.
1	A specified host presentation space short session ID was not valid or was not connected.
2	An error was made in specifying parameters.
9	A system error occurred.
40	Disconnected with asynchronous requests pending.

Notes on Using This Function

1. When a **Disconnect from Structured Fields** function is called, any outstanding asynchronous **Read Structured Fields** (126) or **Write Structured Fields** (127) function requests are returned if the application issues the **Get Request Completion** (125) function call. Use the asynchronous form of this function when cleaning up after issuing a Disconnect call.
2. The **Reset System** (21) function will also free any outstanding asynchronous requests (requests that have not been retrieved by the application using the **Get Request Completion** (125) function).

Disconnect Presentation Space (2)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Disconnect Presentation Space** function drops the connection between your EHLLAPI application program and the host presentation space. Also, if a host presentation space is reserved using the **Reserve** (11) function, it is released upon execution of the **Disconnect Presentation Space** function.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 2	
Data String	NA	
Length	NA	
PS Position	NA	

Return Parameters

Return Code	Explanation
0	The **Disconnect Presentation Space** function was successful.
1	Your program was not currently connected to the host presentation space.
9	A system error was encountered.

Notes on Using This Function

1. After the **Disconnect Presentation Space** function is called, functions that interact with the host-connected presentation space are no longer valid (for example, the **Send Key** (3), **Wait** (4), **Reserve** (11) and **Release** (12) functions).
2. Your EHLLAPI application should disconnect from the host presentation space before exiting.
3. The **Disconnect Presentation Space** function does not reset the session parameters to the defaults. Your EHLLAPI application must call the **Reset System** (21) function to accomplish this.

Disconnect Window Service (102)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Disconnect Window Service** function disconnects the window services connection between the EHLLAPI program and the specified host presentation space window.

Prerequisite Calls

Connect Window Services (101)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 102	
Data String	See the following table	
Length	1	4
PS Position	NA	

Data String Contents

Byte	Definition

Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-4	Reserved

Return Parameters

Return Code	Explanation
0	The **Disconnect Window Service** function was successful.
1	Your program is not connected for Window Services.
9	A system error occurred.

Notes on Using This Function

After the **Disconnect Window Service** function has been called, your application no longer manages the presentation space window.

Before exiting the application, you should request a **Disconnect Window Service** function for all presentation spaces that have been connected for Presentation Manager$^{(R)}$ services. If the application exits with an outstanding connection for window services, the subsystem cancels the outstanding connection.

Find Field Length (32)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Find Field Length** function returns the length of a target field in the connected presentation space. This function can be used to find either protected or unprotected fields, but only in a *field-formatted* host presentation space.

This function returns the number of characters contained in the field identified using the call PS position parameter. This includes all characters from the beginning of the target field up to the character preceding the next attribute byte.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 32	
Data String	See the following table	
Length	NA	NA
PS Position	See note	

Note:

PS Position: Identifies the field within the host presentation space at which to start the **Find**. It can be the PS position of any byte within the field in which you desire the **Find** to start.

The calling 2-character data string can contain:

Code	Explanation
␢␢ or T␢	This field
P␢	The previous field, either protected or unprotected.
N␢	The next field, either protected or unprotected
NP	The next protected field
NU	The next unprotected field
PP	The previous protected field
PU	The previous unprotected field

Note:

The ␢ symbol represents a required blank.

Return Parameters

This function returns a length and a return code.

Length:

The following lengths are valid:

Length	Explanation
= 0	When return code = 28, field length is 0. When return code = 24, host presentation space is not field formatted.
> 0	Required field length in the host presentation space.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Find Field Length** function was successful.
1	Your program is not connected to a host session.
2	A parameter error was encountered.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	No such field was found.
28	Field length of 0 bytes.

Notes on Using This Function

Except when ␢␢ or T␢ is used as the calling data string, if the field found is the same as the field from which the **Find** started, a return code of 24 is returned.

Find Field Position (31)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Find Field Position** function returns the beginning position of a target field in the host-connected presentation space. This function can be used to find either protected or unprotected fields but only in a *field-formatted* host presentation space.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 31	
Data String	See the following table	
Length	NA	NA
PS Position	See note	

Note:

PS Position: Identifies the field within the host presentation space at which to start the **Find**. It can be the PS position of any byte within the field in which you want the **Find** to start.

The calling 2-character data string can contain:

Code	**Explanation**
␢␢ or T␢	This field
P␢	The previous field, either protected or unprotected
N␢	The next field, either protected or unprotected
NP	The next protected field
NU	The next unprotected field
PP	The previous protected field
PU	The previous unprotected field

Note:

The ␢ symbol represents a required blank.

Return Parameters

This function returns a length and a return code.

Length:

The following lengths are valid:

Length	Explanation
= 0	When return code = 28, field length is 0. When return code = 24, host presentation space is not field-formatted.
> 0	Relative position of the requested field from the origin of the host presentation space. This position is defined to be the first position after the attribute byte.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Find Field Position** function was successful.
1	Your program is not connected to a host session.
2	A parameter error was encountered.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	No such field was found.
28	Field length of 0 bytes.

Notes on Using This Function

Except when Ɗ Ɗ or T Ɗ is used as the calling data string, if the field found is the same as the field from which the **Find** started, a return code of 24 is returned.

Free Communications Buffer (124)

3270	*5250*	*VT*
Yes	No	No

The **Free Communications Buffer** function returns to management memory a buffer that is no longer required by the application. The application should free the buffer prior to exiting the system.

Prerequisite Calls

Allocate Communications Buffer (123)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 124	
Data String	See the following table	
Length	Must be 6	Must be 8
PS Position	NA	

Data String Contents

Byte		Definition
Standard	Enhanced	
1-2	1-4	Must be 0
3-6	5-8	The address of the buffer

Return Parameters

Return Code	Explanation
0	The **Free Communications Buffer** function was successful.
2	An error was made in specifying parameters.
9	A system error occurred.
41	The buffer is in use.

Notes on Using This Function

1. If the application attempts to free an in use buffer, the free request will be denied and a return code of 41 will be returned.
2. An application should request the **Free Communications Buffer** (124) function before exiting for all communication buffers that have been allocated using the **Allocate Communications Buffer** (123) function.
3. The **Reset System** (21) function will free buffers allocated by the **Allocate Communications Buffer** (123) function.

Get Key (51)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Get Key** function lets your EHLLAPI application program retrieve a keystroke from a session specified by the **Start Keystroke Intercept** (50) function and either process, accept, or reject that keystroke. By placing this function in a loop, you can use it to intercept a string.

Prerequisite Calls

Start Keystroke Intercept (50)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 51	
Data String	See the following table	
Length	8	12
PS Position	NA	

Data String Contents

Byte		Definition
Standard	Enhanced	
1	1	One of the following values:

		- A 1-character presentation space short name (PSID) - A blank or null indicating a function call for the host-connected presentation
	2-4	Reserved
2-8	5-11	Blanks that hold space for the symbolic representation of the requested data
	12	Reserved

Return Parameters

This function returns a data string and a return code.

Data String:

See the following table:

Byte		Definition
Standard	Enhanced	
1	1	One of the following values: - A 1-character presentation space short name (PSID) - A blank or null indicating a function call for the host-connected presentation
	2-4	Reserved
2	5	An option code character, one of the following characters: - `A` for ASCII returned - `M` for keystroke mnemonic - `S` for special mnemonic
3-8	6-11	These 6 bytes of the preallocated buffer space are used internally to enqueue and dequeue keystrokes. Possible combinations include: - Byte 3 contains an ASCII character and byte 4 contains X'00' - Bytes 3 and 4 contain a double-byte character - Byte 3 contains the escape character (either `@` or another character specified using the `ESC=c` option of function 9) and byte 4 contains a 1-byte abbreviation for a function. (See ASCII Mnemonics) - Bytes 5 through 8 might be similar to bytes 3 and 4 if the returned ASCII mnemonic is longer than 2 bytes (for example, if the ASCII mnemonic represents Attn `@A@Q`, byte 5 contains `@` and byte 6 contains `Q`). If not used, bytes 5 through 8 are set to zero (X'00').

For clarification, some examples of returned data strings are provided below:

Note:

The `@` symbol is the default escape character. The value of the escape character can be set to any keystroke represented in ASCII by using the `ESC=c` option of the **Set Session Parameters** (9) function. If the escape character has been changed to another character using this option, the `@` symbol in the following examples is replaced by the other character.

16-Bit Interface

`EAt`

`E` is the presentation space short name. The keystrokes are returned as ASCII (A), and the returned key is the lowercase letter t. (Bytes 4-8 = X'00').

`EM@2`

`E` is the presentation space short name. The keystrokes are returned as mnemonics, and the returned key is PF2 (Bytes 5-8 = X'00').

32-Bit Interface

`E ÐÐÐAt`

`E` is the presentation space short name. The keystrokes are returned as ASCII (A), and the returned key is the lowercase letter t. (Bytes 7-11 = X'00').

`E ÐÐÐM@2`

`E` is the presentation space short name. The keystrokes are returned as mnemonics, and the returned key is PF2 (Bytes 8-11 = X'00').

Return Code:

The following codes are valid:

Return Code	Explanation
0	The **Get Key** function was successful.
1	An incorrect presentation space was specified.
5	You specified the AID only option under the **Start Keystroke Intercept** (50) function, and non-AID keys are inhibited by this session type when EHLLAPI tries to write incorrect keys to the presentation space.
8	No prior **Start Keystroke Intercept** (50) function was called for this presentation space.
9	A system error was encountered.
20	An undefined key combination was typed.
25	The requested keystrokes are not available on the input queue.
31	Keystroke queue overflowed and keystrokes were lost.

Notes on Using This Function

1. If a return code of 31 occurs for the **Get Key** function, either:
   - Increase the value of the calling length parameter for the **Start Keystroke Intercept** (50) function, or
   - Execute the **Get Key** function more frequently.

   An intercepted keystroke occupies 3 bytes in the buffer. The next intercepted keystroke is placed in the adjacent three bytes. When the **Get Key** function retrieves a keystroke (first in first out, FIFO), the three bytes that it occupied are made available for another keystroke. By increasing the size of the buffer or the rate at which keystrokes are retrieved from the buffer, you can eliminate buffer overflow.

   For the PC/3270, another way to eliminate return code 31 is to operate the PC/3270 emulator in the resume mode.

2. You can use the **Send Key** (3) function

   to pass both original keystrokes and any others that your EHLLAPI application might need to the host-connected presentation space.

3. Keystrokes arrive asynchronously and are enqueued in the keystroke queue that you have provided in your EHLLAPI application program using the **Start Keystroke Intercept** (50) function.
4. The **Get Key** function behaves like a read. When keystrokes are available, they are read into the data area that you have provided in your application.
5. In the case of field support for a session, the application might be interested only in AID keys, for example the Enter key. If so, the **Start Keystroke Intercept** (50) function option code should be set to `D` (meaning for AID Keys only).
6. To use this function, preallocate memory to receive the returned data string parameter. The statements required to preallocate this memory vary depending on the language in which your application is written. Refer to Memory Allocation for more information.

1390/1399 Code Page Support

Unicode functionality is supported only on 3270 and 5250 sessions.

The session option ESC is not supported in a Unicode session; using this option you cannot set a Unicode character as an ESC character. Use the default ESC character @ in a Unicode session. See Set Session Parameters (9) for details.

Prerequisite Calls

Start Keystroke Intercept (50)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 51	
Data String	See the following table	
Length	8	12
PS Position	NA	

Data String Contents

Byte		Definition
Standard	Enhanced	
1	1	One of the following values: • A 1-character presentation space short name (PSID) • A blank or null indicating a function call for the host-connected presentation
	2-4	Reserved
2-8	5-11	Blanks that hold space for the symbolic representation of the requested data
	12	Reserved

Return Parameters

This function returns a data string and a return code.

Data String:

See the following table for 32-bit interface:

Byte	Definition
1	One of the following values: • A 1-character presentation space short name (PSID) • A blank or null indicating a function call for the host-connected presentation
2-4	Reserved
5	`U` is the option code character for a Unicode session.
6-11	The definition of these bytes is similar to the DBCS session; the only difference is that the Unicode character value is stored in bytes 6 and 7 when the option code character is `U`. In a DBCS session, the ASCII character value is stored in byte 3 and byte 4 contains 0X'00' when the option code character is `A`.

Return Code:

The following codes are valid:

Return Code	Explanation
0	The **Get Key** function was successful.
1	An incorrect presentation space was specified.
5	You specified the AID only option under the **Start Keystroke Intercept** (50) function, and non-AID keys are inhibited by this session type when EHLLAPI tries to write incorrect keys to the presentation space.
8	No prior **Start Keystroke Intercept** (50) function was called for this presentation space.
9	A system error was encountered.
20	An undefined key combination was typed.
25	The requested keystrokes are not available on the input queue.
31	Keystroke queue overflowed and keystrokes were lost.

1137 Code Page Support

Unicode functionality is supported only on 5250 sessions.

The session option ESC is not supported in a Unicode session; using this option you cannot set a Unicode character as an ESC character. Use the default ESC character @ in a Unicode session. See Set Session Parameters (9) for details.

Prerequisite Calls

Start Keystroke Intercept (50)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 51	
Data String	See the following table	
Length	8	12
PS Position	NA	

Data String Contents

Byte		Definition
Standard	Enhanced	
1	1	One of the following values: • A 1-character presentation space short name (PSID) • A blank or null indicating a function call for the host-connected presentation
	2-4	Reserved
2-8	5-11	Blanks that hold space for the symbolic representation of the requested data
	12	Reserved

Return Parameters

This function returns a data string and a return code.

Data String:

See the following table for 32-bit interface:

Byte	Definition
1	One of the following values: • A 1-character presentation space short name (PSID) • A blank or null indicating a function call for the host-connected presentation
2-4	Reserved
5	`U` is the option code character for a Unicode session.
6-11	The definition of these bytes is similar to the SBCS session, the only difference is that the Unicode character value is stored in bytes 6 and 7 when the option code character is `U`. In a DBCS session, the ASCII character value is stored in byte 3 and byte 4 contains 0X'00' when the option code character is `A`.

Return Code:

The following codes are valid:

Return Code	Explanation
0	The **Get Key** function was successful.
1	An incorrect presentation space was specified.
5	You specified the AID only option under the **Start Keystroke Intercept** (50) function, and non-AID keys are inhibited by this session type when EHLLAPI tries to write incorrect keys to the presentation space.
8	No prior **Start Keystroke Intercept** (50) function was called for this presentation space.
9	A system error was encountered.
20	An undefined key combination was typed.
25	The requested keystrokes are not available on the input queue.
31	Keystroke queue overflowed and keystrokes were lost.

Get Request Completion (125)

3270	*5250*	*VT*
Yes	No	No

The **Get Request Completion** function allows an application to determine the status of a previous asynchronous function request issued to the EHLLAPI and to obtain the function parameter list before using the data string again. This function is valid only if the user specified asynchronous (A) completion on a previous function call such as **Read Structured Fields** (126) or **Write Structured Fields** (127).

Each asynchronous request requiring the **Get Request Completion** function will return a unique ID from the asynchronous request. The application must save this ID. This ID is the identification used by the **Get Request Completion** function to identify the desired request. The user has three request options using this function:

1. The application can query or wait for a specific asynchronous function request by supplying the request ID of that function and a nonblank session short name.
2. The application can query or wait for the first completed asynchronous function request for a specified session by supplying a request ID of X'0000' and a nonblank session short name.

Prerequisite Calls

Connect Structured Fields (120) and **Allocate Communications Buffer** (123)

and

Read Structured Fields (126) or **Write Structured Fields** (127)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 125	
Data String	See the following table	
Length	Must be 14	Must be 24
PS Position	NA	

Data String Contents

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-4	Reserved
2	5	N or W N=NOWAIT is required W=WAIT is required
	6-8	Reserved
3-4	9-10	Function request ID.
5-6	11-12	Reserved
7-10	13-16	Reserved
11-12	17-20	Reserved
13-14	21-24	Reserved

The **Get Request Completion** function behaves differently depending upon the second character of the parameter string, which is one of the following characters:

N

Nowait option: If a specific request ID was supplied and the function has completed, control will be returned to the application with a return code of zero and a completed data string as defined in Return Parameters. If a request ID of zero was supplied and any eligible asynchronous function has completed, control will be returned to the application with a return code of zero and a completed data string as defined in Return Parameters.

W

Wait option: If a specific request ID was supplied and the function has not completed, the call will wait until the function has completed before returning to the application. If the supplied request ID was zero and no eligible asynchronous function has completed, the call will wait until a function completes before returning to the calling application. On return, the return code value will be zero and the data string will be completed as defined in Return Parameters.

Return Parameters

Byte		Definition
Standard	Enhanced	

5-6	11-12	Function number of the completed asynchronous function (126 or 127). (returned)
7-10	13-16	Address of the data string of the completed asynchronous function call. (The application must not reuse the data string until the request has completed). (returned)
11-12	17-20	Length of the data string of the completed asynchronous function call. (returned)
13-14	21-24	Return code of the completed asynchronous function call. (returned)

Return Code	Explanation
0	The **Get Request Completion** function was successful.
2	An error was made in specifying parameters.
9	A system error was encountered.
38	Requested function was not complete.
42	No matching request was found.

There are some differences between return codes 38 and 42:

1. Return code 38
   a. If a specific request ID and session were requested, both the session and ID were found but the request is pending (not in a completed state).
   b. If a zero request ID and a specific session were requested, the specified session has pending requests, but they are not satisfied (complete).
   c. If a zero request ID and a blank session were requested, pending requests were found but none were satisfied (complete).
2. Return code 42
   a. If a specific request ID and session were requested, the specific request ID was not found in either a pending or a completed state.
   b. If a zero request ID and a specific session were requested, the specific session contains no pending or completed requests.
   c. If a zero request ID and a blank session were requested, no pending or completed requests were found.

Notes on Using This Function

1. This function is valid only if the user specified asynchronous completion (A for Asynchronous) on a previous function call such as **Read Structured Fields** or **Write Structured Fields**.
2. If the return code is a 0, the application should check the returned data string for information pertaining to the completion of the requested asynchronous function.

Lock Presentation Space API (60)

3270	*5250*	*VT*
Yes	No	No

The **Lock Presentation Space API** function allows the application to obtain or release exclusive control of the presentation space window over other Windows 32-bit applications. While locked, no other application can connect to the presentation space window.

Successful processing of this function with the Lock causes EHLLAPI presentation space window functions requested from other EHLLAPI applications to be queued until the requesting application unlocks the presentation space. Requests from the locking application are processed normally.

Prerequisite Calls

Connect to Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 60	
Data String	See the following table	
Length	Must be 3	Must be 8
PS Position	NA	

Data String Contents

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID).
	2-4	Reserved.
2	5	One of the following characters: • L to lock the API. • U to unlock the API.
3	6	One of the following characters: • R to return if the presentation space is already locked by an application. • Q to queue the Lock request if the presentation space is already locked by an application.
	7-8	Reserved.

Return Parameters

Return Code	Explanation
0	The **Lock Presentation Space API** function was successful.
1	An incorrect host presentation space short session ID was specified or was not connected.
2	An error was made in specifying parameters.
9	A system error was encountered.
43	The API was already locked by another EHLLAPI application (on LOCK) or API not locked (on UNLOCK).

Notes on Using This Function

The following EHLLAPI functions are queued when a lock is in effect:

- **Send Key** (3)
- **Copy Presentation Space** (5)
- **Search Presentation Space** (6)
- **Copy Presentation Space to String** (8)
- **Release** (11)
- **Reserve** (12)
- **Query Field Attribute** (14)
- **Copy String to Presentation Space** (15)
- **Search Field** (30)
- **Find Field Position** (31)
- **Find Field Length** (32)
- **Copy String to Field** (33)

- **Copy Field to String** (34)
- **Set Cursor** (40)
- **Send File** (90)
- **Receive File** (91)
- **Connect to Presentation Space** (1) with the CONPHYS parameter set in a previous **Set Sessions Parameter** (9) function call.

These queued requests are not serviced until the lock is removed. When the lock is removed, the queued requests are processed in first-in-first-out (FIFO) order. EHLLAPI functions not listed are run as if there was no lock. The requesting application unlocks the presentation space window by one of the following methods:

- Disconnecting from the presentation space while still owning the Lock.
- Issuing the **Reset System** (21) function while still owning the Lock.
- Stopping the application while still owning the Lock.
- Stopping the session.
- Successfully issuing the **Lock Presentation Space API** with the Unlock option.

Before exiting the application, you should unlock any presentation space windows that have been locked with the **Lock Presentation Space API** function. If the application exits with outstanding locks, or a **Reset System** (21), or **Disconnect Presentation Space** (2) function is issued, the locks are released.

It is recommended that applications lock the presentation space only for short periods of time and only when exclusive use of the presentation space is required.

Lock Window Services API (61)

3270	*5250*	*VT*
Yes	No	No

The **Lock Window Services API** function allows the application to obtain or release exclusive control of the presentation space window over other Windows 32-bit applications. While locked, no other application can connect to the presentation space window.

Successful processing of this function with the Lock causes EHLLAPI presentation space window functions requested from other EHLLAPI applications to be queued until the requesting application unlocks the presentation space. Requests from the locking application are processed normally.

Prerequisite Calls

Connect Window Services (101)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 61	
Data String	See the following table.	
Length	Must be 3	Must be 8
PS Position	NA	

Data String Contents

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID).
	2-4	Reserved.
2	5	One of the following characters: • L to lock the API. • U to unlock the API.
3	6	One of the following characters: • R to return if the presentation space is already locked by an application. • Q to queue the Lock request if the presentation space is already locked by an application.
5-6	11-12	Function number of the completed asynchronous function (126 or 127). (returned)
	7-8	Reserved.

Return Parameters

Return Code	Explanation
0	The **Lock Window Services API** function was successful.
1	An incorrect host presentation space short session ID was specified or was not connected.
2	An error was made in specifying parameters.
9	A system error was encountered.
38	Requested function was not complete.
43	The API was already locked by another EHLLAPI application (on LOCK) or API not locked (on UNLOCK).

Notes on Using This Function

The following EHLLAPI functions are queued when a lock is in effect:

- **Window Status** (104)
- **Change Switch List Name** (105)
- **Change PS Window Name** (106)

These queued requests are not serviced until the lock is removed. When the lock is removed, the queued requests are processed in first-in-first-out (FIFO) order.

The requesting application unlocks the presentation space window by one of the following methods:

- Successfully issuing the **Lock Window Services API** with the UNLOCK option.
- Disconnecting from the presentation space while still owning the Lock.
- Issuing the **Reset System** (21) function while still owning the Lock.
- Stopping the application while still owning the Lock.
- Stopping the session.

Before exiting the application, you should Unlock any presentation space windows that have been locked with the **Lock Window Services API** function. If the application exits with outstanding locks, the subsystem releases the locks.

It is recommended that applications lock the presentation space only for short periods of time and only when exclusive use of the presentation space is required.

Pause (18)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Pause** function waits for a specified amount of time. It should be used in place of *timing loops* to wait for an event to occur. A **Pause** function can be ended by a host event if a prior **Start Host Notification** (23) function has been called and the IPAUSE option is selected.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 18	
Data String	NA	
Length	Contains the pause duration in half-second increments	
PS Position	NA	

Return Parameters

Return Code	Definition
0	The wait duration has expired.
9	An internal system error was encountered. The time results are unpredictable.
26	The host session presentation space or OIA has been updated. Use the **Query Host Update** (24) function to get more information.

Notes on Using This Function

1. Selecting the FPAUSE or IPAUSE option using the **Set Session Parameters** (9) function affects the length of the pause you get when you call this function. See item 6 for more information.
2. The value entered in the calling length parameter is the maximum number of half-second intervals that the **Pause** function waits. For a pause of 20 seconds, a hex value of `0028` (decimal 40) must be passed in the calling length parameter.
3. If you use the IPAUSE option and the pause value is zero, then the function waits up to 2400 half-second intervals, unless interrupted sooner. If you use the FPAUSE option and the pause value is zero, then the function returns immediately.
4. If you use the IPAUSE option, once a pause has been satisfied by a host event, you should call the **Query Host Update** (24) function to clear the queue prior to the next **Pause** function. The **Pause** function will continue to be satisfied with the pending event until the **Query Host Update** (24) function is completed.
5. A practical maximum value for the **Pause** function is `2400`. You should not use the **Pause** function for these kinds of tasks:
   - Delay for very long durations (of several hours, for example).
   - Delay for more than a moderate length of time (20 minutes) before checking the system time-of-day clock and proceeding with your EHLLAPI program execution.
   - With applications requiring a high-resolution timer because the time interval created by a **Pause** function is approximate.
   - Set the time interval to zero in a loop.
6. IPAUSE set and the interruptible pause allow an EHLLAPI application to determine whether the specified host presentation space (PS) or operator information area (OIA) is updated. The following three functions are used:

- **Start Host Notification** (23)
- **Query Host Update** (24)
- **Stop Host Notification** (25)

By using IPAUSE when the **Start** function is called, you can make an application wait until the host presentation space or OIA (or both) receives an update. When the receive is completed and the application can issue the **Query** function to determine the changes, **Pause** terminates. Then the application issues the **Search Presentation Space** (6) to check whether the expected update occurred.

Post Intercept Status (52)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Post Intercept Status** function informs the Personal Communications emulator that a keystroke obtained through the **Get Key** (51) function was accepted or rejected. When the application rejects a keystroke, the **Post Intercept Status** function issues a beep.

Prerequisite Calls

Start Keystroke Intercept (50)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 52	
Data String	See the following table	
Length	Must be 2	Must be 8
PS Position	NA	

The calling data string can contain:

Byte		**Definition**
Standard	Enhanced	
1	1	One of the following values: • The 1-letter short name of the presentation space. • A blank or null indicating a function call for the host-connected presentation space.
	2-4	Reserved
2	5	One of the following characters: • A for accepted keystroke. • R for rejected keystroke.
	6-8	Reserved.

Return Parameters

Return Code	**Explanation**
0	The **Post Intercept Status** function was successful.
1	An incorrect presentation space was specified.

2	An incorrect session option was specified.
8	No prior **Start Keystroke Intercept** (50) function was called for this presentation space ID.
9	A system error was encountered.

Query Additional Field Attribute (45)

3270	***5250***	***VT***
No	Yes	No

The **Query Additional Field Attribute** function returns additional information about the 5250 field containing the input host presentation space position. This information is returned in the data string parameter in the form of a defined structure.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 45.	
Data String	8 bytes long character string.	
Length	8 is implied.	
PS Position	Identifies the target. This can be the PS position of any byte within the target field.	

The calling data string can contain:

Byte	**Definition**
1-8	Reserved

Return Parameters

This function returns a data string and a return code.

Data String:

The function returns the following data string.

Byte	**Definition**
1-6	Reserved
7-8	Two 8-bit unsigned characters that return: • R if field is RTL and L if field is LTR. • U if field is upper case and L if field is a normal case field.

Return Code:

The following return codes are defined:

Return Code	Explanation
0	The **Query Additional Field Attribute** was successful.
1	Your program is not currently connected to a host session.
7	The host presentation space position is not valid.
9	No field was found in this position.
24	Field is unformatted.

Query Close Intercept (42)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Query Close Intercept** function allows the application to determine if the close option was selected.

Prerequisite Calls

Start Close Intercept (41)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 42	
Data String	See the following table.	
Length	Must be 1	Must be 4
PS Position	NA	

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1	1	1-character short session ID of the host presentation space, or a blank or null indicating request for querying the host-connected session
	2-4	Reserved

Return Parameters

Return Code	Explanation
0	A close intercept event did not occur.
1	The presentation source was not valid.
2	An error was made in specifying parameters.
8	No prior **Start Close Intercept** (41) function was called for this host presentation space.
9	A system error occurred.
12	The session stopped.
26	A close intercept occurred since the last query close intercept call.

Query Communications Buffer Size (122)

3270	*5250*	*VT*
Yes	No	No

The **Query Communications Buffer Size** function allows an application to determine both the maximum and the optimum buffer sizes supported by the emulation program.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 122	
Data String	See the following table	
Length	Must be 9	Must be 20
PS Position	NA	

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-4	Reserved
2-3	5-8	16- or 32-bit field for the optimum supported inbound buffer size (Returned value)
4-5	9-12	16- or 32-bit field for the maximum supported inbound buffer size (Returned value)
6-7	13-16	16- or 32-bit field for the optimum supported outbound buffer size (Returned value)
8-9	17-20	16- or 32-bit field for the maximum supported outbound buffer size (Returned value)

Return Parameters

Return Code	Explanation
0	The **Query Communications Buffer Size** function was successful.
1	A specified host presentation space short session ID was not valid or was not connected.
2	An error was made in specifying parameters.
9	A system error occurred.
10	The function was not supported by the emulation program.

Notes on Using This Function

1. There is no way to require the user to use this function. It is not a required function so that the application can be tailored to run on any system.
2. The buffer sizes returned represent the record sizes that are actually transmitted across the medium. For a DDM connection, the 8-byte header supplied in the **Read** and **Write Structured Fields** data buffer is stripped off and 1 byte containing the structured field AID value is prefixed. The application should compare the size of the actual data in the data buffer (which does not include the 8-byte header) with the buffer sizes returned by the **Query Communications Buffer Size** minus 1 byte. For destination/origin connections, the 8-byte header supplied in the **Read** and **Write Structured Fields** data buffer is stripped off and 9 bytes are then prefixed to the data. The application should compare the size of the actual data in the data buffer (which does not include the 8-byte header) with the buffer

size returned by the **Query Communications Buffer Size** minus 9 bytes.

3. The maximum buffer sizes returned represent the maximum number of bytes supported by the workstation hardware and by the emulator. The maximum buffer size can be used only if the host is also configured to accept at least these maximum sizes.
4. The optimum buffer sizes returned represent the optimum number of bytes supported by the both the workstation hardware and the emulator. Some network configurations might set transmission limits smaller than these values. In these cases, the data transfer buffer size override value in the emulator configuration profile will be used for structured field support. The **Query Communications Buffer Size** will reflect any buffer size override values entered in the emulator configuration profile.

Query Communication Event (81)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Query Communication Event** function lets the EHLLAPI program determine whether any communication events have occurred.

Prerequisite Calls

Start Communication Notification (80)

Call Parameters

	Enhanced Interface
Function Number	Must be 81
Data String	1-character short name of the host presentation space or a blank or null indicating request for updates to the host-connected presentation space
Length	4 is implied
PS Position	NA

The calling data structure contains these elements:

Byte	Definition
1	A 1-character presentation space short name (PSID)
2-4	Reserved

Return Parameters

Return Code	Definition
0	The function was successful
1	An incorrect PSID was specified
8	No prior call to **Start Communication Notification** (80) function was called for the PSID
9	A system error was encountered
21	The indicated PSID was connected
22	The Indicated PSID was disconnected

Query Cursor Location (7)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Query Cursor Location** function indicates the position of the cursor in the host-connected presentation space by returning the cursor position.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 7	
Data String	NA	
Length	NA	
PS Position	NA	

Return Parameters

This function returns a length and a return code.

Length:

Host presentation space position of the cursor.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Query Cursor Location** function was successful.
1	Your program is not currently connected to a host session.
9	A system error was encountered.

Query Field Attribute (14)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Query Field Attribute** function returns the attribute byte of the field containing the input host presentation space position. This information is returned in the returned length parameter.

For the PC/3270, note also that:

- The returned length parameter is set to 0 if the screen is unformatted.
- Attribute bytes are equal to or greater than hex C0.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 14.	
Data String	NA.	
Length	NA.	
PS Position	Identifies the target. This can be the PS position of any byte within the target field.	

Return Parameters

This function returns a length and a return code.

Length:

The attribute value if the screen is formatted, or 0 if the screen is unformatted.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Query Field Attribute** was successful.
1	Your program is not currently connected to a host session.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	Attribute byte not found or unformatted host presentation space.

Notes on Using This Function

The returned field attributes are defined in the following tables. The bit positions are in IBM format with bit 0 as the left most bit in the byte.

- 3270 field attribute:

Bit Position	Meaning
0-1	Both = 1, field attribute byte
2	Unprotected/protected 0 = Unprotected data field 1 = Protected field
3	A/N 0 = Alphanumeric data 1 = Numeric data only

4-5	I/SPD 00 = Normal intensity, pen not detectable 01 = Normal intensity, pen detectable 10 = High intensity, pen detectable 11 = Nondisplay, pen not detectable
6	Reserved
7	MDT 0 = Field has not been modified 1 = Field has been modified

- 5250 field attributes:

Bit Position	**Meaning**
0	Field attribute flag 0 = Nonfield attribute flag 1 = Field attribute flag
1	Visibility 0 = Nondisplay 1 = Display
2	Unprotected/protected 0 = Unprotected data field 1 = Protected field
3	Intensity 0 = Normal intensity 1 = High intensity
4-6	Field type 000 = Alphanumeric data: All characters are available 001 = Alphabet only: Uppercase and lowercase, comma, period, hyphen, blank, or Dup key are available 010 = Numeric shift: Automatic shift for number 011 = Numeric data only: 0-9, comma, period, plus, minus, blank, or Dup key are available 101 = Numeric data only: 0-9, or Dup key are available 110 = Magnetic stripe reading device data only 111 = Signed-numeric data: 0-9, plus, minus, or Dup key are available
7	MDT 0 = Field has not been modified

	1 = Field has been modified

Query Host Update (24)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Query Host Update** function lets the programmed operator determine if the host has updated the host presentation space or OIA because:

- The **Start Host Notification** (23) function was called

  (on first call to the **Query Host Update** function only)

- The previous call to the **Query Host Update** function (for all calls to the **Query Host Update** function except the first).

Prerequisite Calls

Start Host Notification (23)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 24	
Data String	1-character short name of the host presentation space, or a blank or null indicating request for updates to host-connected presentation space	
Length	1 is implied	4 is implied
PS Position	NA	

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-4	Reserved

Return Parameters

Return Code	Definition
0	No updates have been made since the last call.
1	An incorrect host presentation space was specified.
8	No prior **Start Host Notification** (23) function was called for the host presentation space ID.
9	A system error was encountered.
21	The OIA was updated.
22	The presentation space was updated.
23	Both the OIA and the host presentation space were updated.
44	Printing has completed in the printer session.

Notes on Using This Function

The target presentation space must be specified in the data string, even though a connection to the host presentation space is not necessary to check for updates.

Query Session Status (22)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Query Session Status** function is used to obtain session-specific information.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	16-bit	**32-bit**
Function Number	Must be 22.	
Data String	An 18/20-byte string consisting of a 1-byte short name of the target presentation space plus 17 bytes for returned data. Position 1 can be filled with: 1. A blank or a null to indicate a request for the host_connected presentation space. 2. An * (asterisk) to indicate a request for the keyboard-owner presentation space.	
Length	Must be 18	Must be 20
PS Position	NA	

Return Parameters

This function returns a data string and a return code.

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-4	Reserved
2-9	5-12	Session long name (same as profile name; or, if profile not set, same as short name)
10	13	**Session** **Type** **D** 3270 display **E** 3270 printer **F** 5250 display **G** 5250 printer **H**

		ASCII VT
11	14	Session characteristics expressed by a binary number including the following session-characteristics bits **Bit 0** EAB 0: Session has the basic attribute. 1: Session has the extended attribute **Bit 1** PSS 0: Session does not support the programmed symbols 1: Session supports the programmed symbols **Bits 2-7** Reserved
12-13	15-16	Number of rows in the host presentation space, expressed as a binary number
14-15	17-18	Number of columns in the host presentation space, expressed as a binary number
16-17	19-20	Host code page expressed as a binary number
18		Reserved

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Query Session Status** function was successful.
1	An incorrect host presentation space was specified.
2	An incorrect string length was made.
9	A system error was encountered.

Notes on Using This Function

1. To use this function, preallocate memory to receive the returned data string parameter. The statements required to preallocate this memory vary depending on the language in which your application is written. See Memory Allocation for more information.

Query Sessions (10)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Query Sessions** function returns a 16-byte (12-byte for standard interface) data string describing each host session.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

Function	Description	
	Standard Interface	Enhanced Interface
Function Number	Must be 10	

Data String	Preallocated string of 16*n* bytes long (12*n* for 16-bit) (*n* =number of sessions) or more	
Length	12*n* bytes	16*n* bytes
PS Position	NA	

Note:

When the length is not matched to the number of sessions, the return code is 2.

Return Parameters

This function returns a data string, a length, and a return code.

Data String:

The returned data string is 16*n* bytes long (12*n* for standard interface), where *n* is the number of host sessions. The descriptors are concatenated into the data string and each session type, and presentation space size of a host session.

The format of each 16-byte (12-byte for standard interface) session descriptor is as follows:

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-4	Reserved
2-9	5-12	Session long name (same as profile name; or, if profile not set, same as short name)
10	13	Connection type H=host
	14	Reserved
11-12	15-16	Host presentation space size (this is a binary number and is not in display format). If the session type is a print session, the value is 0.

Length:

The number of host sessions started.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Query Sessions** function was successful.
2	An incorrect string length was made.
9	A system error was encountered.

Notes on Using This Function

1. If an application program receives RC=2 or RC=0, the number of the active sessions is returned in the length field. The application program can recognize the minimum string length by this number.
2. The **Query Sessions** function is affected by the CFGSIZE/NOCFGZISE session option (see item 16 for more information) and by the EXTEND_PS/NOEXTEND_PS option (see item 22 for more information).

Notes:

1. When NOCFGSIZE is set in **Set Session Parameters** (9) for a 5250 session, the value of presentation space size returned in byte position 11 and 12 from **Query Sessions**(10) will be changed in accordance with the selection of EXTEND_PS or NOEXTEND_PS.
2. When EXTEND_PS is set in **Set Session Parameters** (9), presentation space size returned from **Query Sessions** (10) will include the size of the message line, if it exists.
3. When NOEXTEND_PS is set, the value will not change regardless of the existence of a message line. In the case of 25 row, 80 column presentation space, the value can be 1920 or 2000.

Query System (20)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Query System** function can be used by an EHLLAPI application program to determine the level of Personal Communications support and other system-related values. This function returns a string that contains the appropriate system data. Most of this information is for use by a service coordinator when you call the IBM Support Center after receiving a return code 9

(a system error was encountered).

The bytes in this returned string are defined in Return Parameters.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 20	
Data String	Preallocated string of 35 bytes	36 bytes
Length	Must be 35	Must be 36
PS Position	NA	

Return Parameters

This function returns a data string and a return code.

Data String:

A data string of 35 bytes (for 16-bit) or 36 bytes (for 32-bit) is returned. The bytes are defined as follows:

Byte		Definition
Standard	Enhanced	
1	1	EHLLAPI version number
2-3	2-3	EHLLAPI level number
4-9	4-9	Reserved
10-12	10-12	Reserved
13	13	Hardware base, U=Unable to determine
14	14	Program type, where P=IBM Personal Communications
15-16	15-16	Reserved
17-18	17-18	Personal Communications version/level as a 2-byte ASCII value
19	19	Reserved
20-23	20-23	Reserved
24-27	24-27	Reserved
28-29	28-29	Reserved
	30	Reserved
30-31	31-32	NLS type expressed as a 2-byte binary number
33-35	34-36	Reserved

Return Code

The following codes are defined:

Return Code	Explanation
0	The **Query System** function was successful; data string has been returned.
1	EHLLAPI is not loaded. (PC/3270 only)
2	An incorrect string length was specified. (PC/3270 only)
9	A system error was encountered.

Notes on Using This Function

To use this function, preallocate memory to receive the returned data string parameter. See Memory Allocation for more information.

Query Window Coordinates (103)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Query Window Coordinates** function requests the coordinates for the window of a presentation space. The window coordinates are returned in pels.

Note:

(0,0) indicates the top-left of the window.

Prerequisite Calls

Connect Window Services (101)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 103	
Data String	1-character short session ID of the host presentation space	
Length	17 is implied	20 is implied
PS Position	NA	

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1	1	One of the following values: • A 1-character presentation space short name (PSID) • A blank or null indicating a function call for the current connection presentation space
	2-4	Reserved
2-17	5-20	Reserved

Return Parameters

This function returns a data string and a return code.

Byte		Definition
Standard	Enhanced	
1	1	One of the following values: • A 1-character presentation space short session ID • A blank or null indicating a function call for the current connection presentation space
	2-4	Reserved
2-17	5-20	Four 32-bit unsigned integers that return:
2-5	5-8	XLeft Long integer in pels of the left X coordinate of the rectangular window relative to the desktop window
6-9	9-12	YBottom Long integer in pels of the bottom Y coordinate of the rectangular window relative to the desktop window
10-13	13-15	XRight Long integer in pels of the right X coordinate of the rectangular window relative to the desktop window
14-17	16-20	YTop Long integer in pels of the top Y coordinate of the rectangular window relative to the desktop window

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Query Window Coordinates** function was successful.
1	Your program was not currently connected to the host session.
9	A system error occurred.
12	The session stopped.

Read Structured Fields (126)

3270	*5250*	*VT*
Yes	No	No

The **Read Structured Fields** function allows an application to read structured field data from the host application. If the call specifies S (for Synchronous), the application does not receive control until the **Read Structured Fields** is completed. If the call specifies A (for Asynchronous), the application receives control immediately after the call. If the call specifies M (for Asynchronous, message mode), the application receives control immediately after the call. The application can wait for the message. In any case (S, A, or M), the application provides the buffer address in which the data from the host is to be placed.

For a successful asynchronous completion of this function, the following statements apply:

The return code field in the parameter list might not contain the results of the requested I/O. If the return code is not 0, the request failed. The application must take the appropriate action based on the return code.

If the return code for this request is 0, the application must use the request ID returned with this function call to issue the **Get Request Completion** function call to determine the completion results of the function associated with the request ID. The **Get Request Completion** function call returns the following information:

1. Function request ID
2. Address of the data string from the asynchronous request
3. Length of the data string
4. Return code of the completed function

Prerequisite Calls

Connect for Structured Fields (120) and **Allocate Communication Buffer** (123)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 126	
Data String	See the following table	
Length	8, 10 or 14	20
PS Position	NA	

The calling data string can contain:

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID).
	2-4	Reserved.
2	5	S or A or M **S =** Synchronous. Control is not returned to the application until the read is satisfied. **A =** Asynchronous. Control is returned immediately to the application, can wait for the event object. **M =** Asynchronous. Control is returned immediately to the application, can wait for the message.
	6	Reserved.
3-4	7-8	2-byte destination/origin ID.
5-8	9-12	4-byte address of the buffer into which the data is to be read. The buffer must be obtained using the **Allocate Communications Buffer** (123) function.

9-10	13-16	Reserved.
11-12	17-20	When M is specified in position 2 the window handle of the window that receives the message should be set. The message is a return value of RegisterWindowMessage ("PCSHLL")(not equal 0).
13-14		The data in these positions is ignored by EHLLAPI. However, no error is caused if the migrating program has data in these positions. This data is accepted to provide compatibility with migrating applications.

Return Parameters

This function returns a data string and a return code.

Data String:

If A (asynchronous) is specified in position 5, (2 for standard interface) and the function is completed successfully, the following data string is returned:

Byte		Definition
Standard	Enhanced	
9-10	13-14	2-byte function request ID. It is used by the **Get Request Completion** (125) function to determine the completion of this function call.
	15-16	Reserved.
	17-20	4-byte value in which the event object address is returned by EHLLAPI. The application can wait for this event object. When the event object is cleared, the application must issue the **Get Request Completion** (125) function call (32-bit only).

Note:

A event object address is returned for each successful asynchronous request. The event object should not be used again. A new event object is returned for each request and is valid for only the duration of that request.

Data String:

If "M" (asynchronous message mode) is specified in position 5 (2 for 16-bit applications) and the function is completed successfully, the following data string is returned:

Byte		Definition
9-10	13-14	A 2-byte function request ID. It is used by the **Get Request Completion (125)** function to determine the completion of this function call.
	15-16	Reserved.
11-12	17-18	Task ID of asynchronous message mode.
	19-20	Reserved.

Note:

If the function is completed successfully, an application window receive a message. The message is a return value of RegisterWindowMessage (PCSHLL). The wParam parameter contains Task ID returned by the function call. The HIWORD of lParam parameter contains Return Code 0, which shows the function was successful, and LOWORD of lParam parameter contains function number 126.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Read Structured Fields** function was successful.
1	A specified host presentation space short session ID was not valid or was not connected.
2	An error was made in specifying parameters.
9	A system error occurred.
11	Resource unavailable (memory unavailable).
35	Request rejected. An outbound transmission from the host was canceled.
36	Request rejected. Lost contact with the host.
37	The function was successful, but the host is inbound disabled.

Notes on Using This Function

1. Return code 35 will be returned when the first **Read Structured Fields** or **Write Structured Fields** is requested after an outbound transmission from the host is canceled. Corrective action is the responsibility of the application.
2. Return code 36 requires that the application disconnect from the emulation program and then reconnect to reestablish communication with the host. Corrective action is the responsibility of the application.
3. Return code 37 will be returned if the host is inbound disabled. The **Read Structured Fields** function was successfully requested.
4. The EHLLAPI allows for a maximum of 20 asynchronous requests per application to be outstanding. A return code for unavailable resources (RC=11) is returned if more than 20 asynchronous requests are attempted.
5. If you are using an IBM Global Network(R) connection, the maximum number of asynchronous requests is 10.

The structured field data contains the application structured fields received from the host. Structured field headers are removed by the EHLLAPI before the structured field data reaches the application.

The structured field data format is as follows:

Offset	Length	Contents
0	1 word	X'0000'.
2	1 word	m (message length: The number of bytes of data in the message, the number does not include the buffer header prefix, which contains 8 bytes). This value is returned by EHLLAPI.
4	1 word	n (buffer size: the supplied length of the data buffer that does include the 8-byte message header). This value must be set by the application.
6	1 word	X'C000'.
8	8 bytes	Length of the first (or only) structured field message.
10	1 byte	First nonlength byte of the structured field message.
		. . .
m+7	1 byte	Last byte in the structured field message.

Bytes 0 through 7 are the buffer header. These first 8 bytes are used by the emulation program. The user section of the buffer begins with offset 8. Bytes 8 and 9 contain the number of bytes in the first structured field (a structured field message can contain multiple structured fields), including 2 bytes for bytes 8 and 9. Bytes 8 through *m*+7 are used for the structured field message received from the host (which could contain multiple structured fields).

The using application must furnish the complete buffer with the word at offset 0 set to zero. The buffer length must be in the word at offset 4. The word at offset 6 must be X'C000'. The emulation program will place the data message beginning at offset 8 and place the length of the message in the word at offset 2. The buffer length is not disturbed by EHLLAPI.

Synchronous Requests

When **Read Structured Fields** is requested synchronously (the S option in the data string), control is returned to the application only after the request is satisfied. The application can assume:

- The return code is correct.
- The data in the communications buffer (read buffer) is correct.
- The host is no longer processing the **Read Structured Fields** request.

Asynchronous Requests

When **Read Structured Fields** is requested asynchronously (the A option in the data string), the application *cannot* assume:

- The return code is correct.
- The data in the communications buffer (read buffer) is correct.
- The host is no longer processing the **Read Structured Fields** request.

When requested asynchronously, EHLLAPI returns the following values:

- A 16-bit Request ID in positions 13-14 (9-10 for standard interface) of the data string
- The address of a event object in positions 17--20 of the data string

These are used to complete the asynchronous **Read Structured Fields** call.

The following steps must be completed to determine the outcome of an asynchronous **Read Structured Fields** function call:

- If the EHLLAPI return code is not zero, the request failed. No asynchronous request has been made. The application must take appropriate actions before attempting the call again.
- If the return code is zero, the application should wait until the event object is in the signaled state by using the **Get Request Completion** (125) function or **Wait For Single Object**. The event object should not be reused. The event object is valid only for the duration of the **Read Structured Fields** function call through the completion of the **Get Request Completion** (125) function call.
- Once the event object is in the signaled state, use the returned 16-bit Request ID as the Request ID parameter in a call to the **Get Request Completion** (125) function. The data string returned from the **Get Request Completion** (125) function call contains the final return code of the **Read Structured Fields** function call.

When **Read Structured Fields** is requested asynchronously (the M option in the data string), the application *cannot* assume:

- The return code is correct.
- The data in the communications buffer (read buffer) is correct.
- The host is no longer processing the **Read Structured Fields** request.

When requested asynchronously with the M option, EHLLAPI returns the following values:

- A 16-bit Request ID in positions 13-14 (9-10 for standard interface) of the data string
- Task ID of asynchronous message mode in positions 17-18 (11-12 for standard interface) of the data string.

These are used to complete the asynchronous **Read Structured Fields** call.

Receive File (91)

3270	*5250*	*VT*
Yes	Yes	No

The **Receive File** function is used to transfer a file from the host session to the workstation session. It is used the same way as the RECEIVE command is used in the PC/3270. The **Receive File** function can be called by an EHLLAPI application program.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 91.	
Data String	Refer to the examples.	
Length	Length, in number of bytes, of the data string. Overridden if in EOT mode.	

Following are examples of the data strings for a single-byte character set (SBSC):

3270 Session

- To receive the file from the VM/CMS host system:

```
pc_filename [id:]fn ft [fm] [(option]
```

- To receive the file from the MVS(TM)/TSO host system:

```
pc_filename[id:]dataset[(member)] [/password] [option]
```

- To receive the file from the CICS(R) host system:

```
pc_filename [id:]host_filename [(option]
```

5250 Session

- To receive the file from the iSeries, eServer i5, or System i5 host system:

```
pc_filename [id:]library file member [option]
```

Following are examples of the data strings for a double-byte character set (DBCS):

3270 Session

- To receive the file from the VM/CMS host system:

```
pc_filename [id:]fn ft [fm]  [(option]
```

- To receive the file from the MVS/TSO host system:

```
pc_filename [id:]dataset[(member)] [/password]
[(option]
```

- To receive the file from the CICS host system:

```
pc_filename [id:]host_filename [(option]
```

5250 Session

- To receive the file from the iSeries, eServer i5, or System i5 host system:

```
pc_filename [id:]library file member [option]
```

Note:

Parameters within [ ] are optional. Available options are listed below.

Host System	Common Options
VM/CMS	ASCII, JISCII, CRLF, APPEND, TIME n, CLEAR, NOCLEAR, PROGRESS, QUIET
MVS/TSO	ASCII, JISCII, CRLF, APPEND, TIME (n), CLEAR, NOCLEAR, PROGRESS, QUIET, AVBLOCK|TRACKS|CYLINDERS
CICS	ASCII, JISCII, CRLF, NOCRLF, BINARY, TIME n, CLEAR, NOCLEAR, PROGRESS, QUIET
i5/OS(TM) or OS/400(R)	ASCII, JISCII, CRLF, APPEND, TIME n, CLEAR, NOCLEAR, PROGRESS, QUIET

Note:

JISCII is valid in a DBCS session for Japan only and ASCII is valid for all other SBCS and DBCS sessions.

Other options specified will be passed to the host transfer program. The file transfer program on the host side either uses them, ignores them, or returns an error. Consult the host transfer program documentation to see a complete list of the options supported.

Return Parameters

Return Code	Explanation
2	Parameter error or you have specified a length that is too long (more than 255 bytes) for the EHLLAPI buffer. The file transfer was unsuccessful.
3	File transfer complete.
4	File transfer complete with segmented records.

9	A system error was encountered.
27	File transfer terminated because of either a Cancel button or the timeout set by the **Set Session Parameter** (9) function.
101	File transfer was successful (transfer to/from CICS).

If you receive return code 2 or 9, there is a problem with the system or with the way you specified your data string.

Other return codes can also be received, which relate to message numbers generated by the host transfer program. For transfers to a CICS host transfer program, subtract 100 from the return code to give you the numeric portion of the message. For example, a return code of 101 would mean that the message number INW0001 was issued by the host. For other host transfer programs, just use the return code as the numerical part of the message. For example, a return of 34 would mean that message TRANS34 was issued by the host transfer program. The documentation for your host transfer program should give more information about the meanings of the specific messages.

Operating system error codes reported by EHLLAPI are greater than 300. To determine the error code, subtract 300 and refer to the operating system documentation for return codes.

Notes on Using This Function

1. Four sets of parameters under the **Set Session Parameters** (9) function are related to this function. They are the STRLEN/STREOT, EOT=c, QUIET/NOQUIET and the TIMEOUT=c/TIMEOUT=0 session options. See items 1 and 2 and items 7 and 8 for more information.
2. If no path is specified when the **Receive File** function is executed, the received file is stored in the current subdirectory, which is the directory in which your application is running.

Release (12)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Release** function unlocks the keyboard that is associated with the host presentation space reserved using the **Reserve** (11) function.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 12	
Data String	NA	
Length	NA	
PS Position	NA	

Return Parameters

Return Code	Explanation
0	The **Release** function was successful.
1	Your program is not connected to a host session.

9	A system error was encountered.

Notes on Using This Function

If you do not **Release** a host presentation space reserved by using the **Reserve** (11) function, you are locked out of that session until you call the **Reset System** (21) function, you call the **Disconnect Presentation Space** (2) function, or you terminate the EHLLAPI application program.

Reserve (11)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Reserve** function locks the keyboard that is associated with the host-connected presentation space to block input from the terminal operator.

The reserved host presentation space remains locked until one of the following occurs:

- **Connect** (1) function is executed to a new session.
- **Disconnect Presentation Space** (2) function is executed.
- **Release** (12) function is executed.
- **Reset System** (21) function is executed.
- **Start Keystroke Intercept** (50) function is executed.
- EHLLAPI application program is terminated.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 11	
Data String	NA	
Length	NA	
PS Position	NA	

Return Parameters

Return Code	Explanation
0	The **Reserve** function was successful.
1	Your program is not connected to a host session.
5	Presentation space cannot be used.
9	A system error was encountered.

Notes on Using This Function

1. If your EHLLAPI application program is sending a series of transactions to the host, you might need to prevent the user from gaining access to that session until

your application processing is complete.

2. The keyboard input that a user makes while the keyboard is locked by this function is enqueued and processed after the session is terminated.
3. This function locks both the mouse and the keyboard input. The application program must unlock the presentation space to enable either the mouse or the keyboard input.

Reset System (21)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Reset System** function reinitializes EHLLAPI to its starting state. The session parameter options are reset to their defaults. Event notification is stopped. The reserved host session is released. The host presentation space is disconnected. Keystroke intercept is disabled.

You can use the **Reset System** function during initialization or at program termination to reset the system to a known initial condition.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 21	
Data String	NA	
Length	NA	
PS Position	NA	

Return Parameters

Return Code	Definition
0	The **Reset System** function was successful.
1	EHLLAPI is not loaded.
9	A system error was encountered.

Notes on Using this Function

For the PC/3270, this function can be used to check whether EHLLAPI is loaded. Place a call to this function at the start of your application and check for a return code of 1.

Search Field (30)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Search Field** function examines a field within the connected host presentation space for the occurrence of a specified string. If the target string is found, this function returns the decimal position of the string numbered from the beginning of the host presentation space. (For example, in a 24-row by 80-column presentation space, the row 1, column 1 position is numbered *1* and the row 5, column 1 position is numbered *321*.)

This function can be used to search either protected or unprotected fields, but only in a *field-formatted* host presentation space.

Note:

If the field at the end of the host presentation space wraps, wrapping occurs when the end of the presentation space is reached.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 30.	
Data String	Target string for search.	
Length	Length of the target data string. Overridden in EOT mode.	
PS Position	Identifies the target field. For `SRCHALL`, this can be the PS position of any byte within the target field. For `SRCHFROM`, it is the beginning point of the search for `SRCHFRWD` or the ending point of the search for `SRCHBKWD`. See note 3.	

Return Parameters

This function returns a length and a return code.

Length:

The following codes are defined:

Length	Explanation
= 0	The string was not found.
> 0	The string was found at the indicated host presentation space position.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Search Field** function was successful.
1	Your program is not connected to a host session.
2	Parameter error. Either the string length was zero, or EOT mode was specified but no EOT character was found in calling data string.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	The search string was not found, or the host presentation space was unformatted.

Notes on Using This Function

1. Four sets of parameters under the **Set Session Parameters** (9) function are related to this function. They are the SRCHALL/SRCHFROM, STRLEN/STREOT, SRCHFRWD/SRCHBKWD, and the EOT=c session options. See items 1 through 4 for more information.
2. You can use the **Set Session Parameters** (9) function to determine whether your searches proceed forward (SRCHFRWD) or backward (SRCHBKWD) in a field.
3. The **Search Field** function normally checks the entire field (SRCHALL default mode). However, you can use the function 9 to specify SRCHFROM. In this mode, the calling PS position parameter does more than identify the target field. It also provides a beginning or ending point for the search.
   - If the SRCHFRWD option is in effect, the search for the designated string begins at the specified PS position and proceeds toward the end of the field.
   - If the SRCHBKWD option is in effect, the search for the designated string begins at the end of the field and proceeds backward toward the specified PS position. If the target string is not found, the search ends at the PS position specified in the calling PS position parameter.
4. **DBCS Only:** If the start position of the specified search function is the second byte in a double-byte character, the search is started from the next character for SRCHFRWD and from the character for SRCHBKWD. If the last character of the specified string is the first byte of a double-byte character, the character is not searched for.

   The search ignores a pair of SO and SI in the presentation space. When you search a double-byte control character, put SO (X'0E') before the character and SI (X'0F') after it. For example, X'0E000C0F' in the data string is treated as a double-byte character FF (X'000C').

Note:

5250 emulation supports a presentation space of 24 rows by 80 columns. In some instances, Communication Manager 5250 emulation displays a 25th row. This occurs when either an error message from the host is displayed or when the operator selects the SysReq key. Personal Communications displays 25th row information on row 24, or on the status bar. For information to be displayed on the status bar, the status bar must be configured. Refer to *Quick Beginnings* for information on configuring the status bar. By the **EXTEND_PS** option, an EHLLAPI application can use the same interface with Communication Manager EHLLAPI and valid presentation space is extended when this condition occurs.

1390/1399 Code Page Support

Unicode functionality is supported only on 3270 and 5250 sessions.

STREOT option is not supported in a Unicode session. Please see Set Session Parameters (9) for details.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 30.	
Data String	Target Unicode string for searching.	
Length	Length of the target Unicode string in Unicode characters. **Note:** The EOT mode is not supported in a Unicode session; therefore, length should be specified for proper functioning of this function in a Unicode session.	
PS Position	Identifies the target field. For `SRCHALL`, this can be the PS position of any byte within the target field. For `SRCHFROM`, it is the beginning point of the search for `SRCHFRWD` or the ending point of the search for `SRCHBKWD`. See note 3.	

Return Parameters

This function returns a length and a return code.

Length:

The following codes are defined:

Length	Explanation
= 0	The string was not found.
> 0	The string was found at the indicated host presentation space position.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Search Field** function was successful.
1	Your program is not connected to a host session.
2	Parameter error. Either the string length was zero, or EOT mode was specified but no EOT character was found in calling data string.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	The search string was not found, or the host presentation space was unformatted.

Notes on Using This Function

The following options are supported in a Unicode session for **Search Field** and function in the same way as in DBCS:

- STRLEN
- SRCHALL
- SRCHFROM
- SRCHFRWD
- SRCHBKWD

1137 Code Page Support

Unicode functionality is supported only on 5250 sessions.

STREOT option is not supported in a Unicode session. Please see Set Session Parameters (9) for details.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 30.	
Data String	Target Unicode string for search.	
Length	Length of the target Unicode string in bytes. **Note:** The EOT mode is not supported in a Unicode session; therefore, length should be specified for proper functioning of this function in a Unicode session.	
PS Position	Identifies the target field. For `SRCHALL`, this can be the PS position of any byte within the target field. For `SRCHFROM`, it is the beginning point of the search for `SRCHFRWD` or the ending point of the search for `SRCHBKWD`. See note 3.	

Return Parameters

This function returns a length and a return code.

Length:

The following codes are defined:

Length	Explanation
= 0	The string was not found.
> 0	The string was found at the indicated host presentation space position.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Search Field** function was successful.
1	Your program is not connected to a host session.
2	Parameter error. Either the string length was zero, or EOT mode was specified but no EOT character was found in calling data string.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	The search string was not found, or the host presentation space was unformatted.

Notes on Using This Function

The following options are supported in a Unicode session for **Search Field** and function in the same way as in SBCS:

- STRLEN
- SRCHALL
- SRCHFROM
- SRCHFRWD
- SRCHBKWD

Search Presentation Space (6)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Search Presentation Space** function lets your EHLLAPI program examine the host presentation space for the occurrence of a specified string.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 6.	
Data String	Target string for search.	
Length	Length of the target data string. Overridden in EOT mode.	
PS Position	Position within the host presentation space where the search is to begin (SRCHFRWD option) or to end (SRCHBKWD option). Overridden in SRCHALL (default) mode.	

Return Parameters

This function returns a length and a return code.

Length:

The following codes are defined:

Length	Explanation
= 0	The string was not found.
> 0	The string was found at the indicated host presentation space position.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Search Presentation Space** function was successful.
1	Your program is not connected to a host session.
2	An error was made in specifying parameters.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	The search string was not found.

Notes on Using This Function

1. Four sets of parameters under the **Set Session Parameters** (9) function are related to this function. They are the SRCHALL/SRCHFROM, STRLEN/STREOT, SRCHFRWD/SRCHBKWD, and the EOT=c session options. See items 1 through 4 through for more information.
2. You can use the **Set Session Parameters** (9) function to specify SRCHBKWD. When this option is in effect, the search operation locates the *last* occurrence of the string.
3. The **Search Presentation Space** function normally checks the entire host presentation space. However, you can use the **Set Session Parameters** (9) function to specify SRCHFROM. In this mode, the calling PS position parameter specifies a beginning or ending point for the search.
   - If the SRCHFRWD option is in effect, the search for the designated string begins at the specified PS position and proceeds toward the end of the host presentation space.
   - If the SRCHBKWD option is in effect, the search for the designated string begins at the end of the PS and proceeds backward toward the specified PS position. If the target string is not found, the search ends at the PS position specified in the calling PS position parameter.
4. The SRCHFROM option is also useful if you are looking for a keyword that might occur more than once in the host presentation space.
5. The **Search Presentation Space** function is useful in determining when the host presentation space is available. If your EHLLAPI application is expecting a specific prompt or message before sending data, the **Search Presentation Space** function allows you to check for a prompt message before continuing.
6. **DBCS Only:** If the start position of the specified search function is the second byte in a double-byte character, the search is started from the next character for SRCHFRWD and from the character for SRCHBKWD. If the last character of the specified string is the first byte of a double-byte character, the character is not searched for.

   The search ignores a pair of SO and SI in the presentation space. When you search a double-byte control character, put SO (X'0E') before the character and SI (X'0F') after it. For example, X'0E000C0F' in the data string is treated as a double-byte character FF (X'000C').

Note:

5250 emulation supports a presentation space of 24 rows by 80 columns. In some instances, Communication Manager 5250 emulation displays a 25th row. This occurs when either an error message from the host is displayed or when the operator selects the SysReq key. Personal Communications displays 25th row information on row 24, or on the status bar. For information to be displayed on the status bar, the status bar must be configured. Refer to *Quick Beginnings* for information on configuring the status bar. By the **EXTEND_PS** option, an EHLLAPI application can use the same interface with Communication Manager EHLLAPI and valid presentation space is extended when this condition occurs.

1390/1399 Code Page Support

Unicode functionality is supported only on 3270 and 5250 sessions.

STREOT option is not supported in a Unicode session. Please refer to Set Session Parameters (9) for details.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 6.	
Data String	Target Unicode string for search.	
Length	Length of the target Unicode string in Unicode characters. **Note:**	

	The EOT mode is not supported in a Unicode session; therefore, length should be specified for proper functioning of this function in a Unicode session.
PS Position	Position within the host presentation space where the search is to begin (SRCHFRWD option) or to end (SRCHBKWD option). Overridden in SRCHALL (default) mode.

Return Parameters

This function returns a length and a return code.

Length:

The following codes are defined:

Length	Explanation
= 0	The string was not found.
> 0	The string was found at the indicated host presentation space position.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Search Presentation Space** function was successful.
1	Your program is not connected to a host session.
2	An error was made in specifying parameters.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	The search string was not found.

Notes on Using This Function

The following options are supported in a Unicode session for **Search Presentation Space** (6) and function in the same way as in DBCS:

- STRLEN
- SRCHALL
- SRCHFROM
- SRCHFRWD
- SRCHBKWD

1137 Code Page Support

Unicode functionality is supported only on 5250 sessions.

STREOT option is not supported in a Unicode session. Please refer to Set Session Parameters (9) for details.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 6.	
Data String	Target Unicode string for search.	
Length	Length of the target Unicode data string in bytes. **Note:** The EOT mode is not supported in a Unicode session; therefore, length should be specified for proper functioning of this function in a Unicode session.	
PS Position	Position within the host presentation space where the search is to begin (SRCHFRWD option) or to end (SRCHBKWD option). Overridden in SRCHALL (default) mode.	

Return Parameters

This function returns a length and a return code.

Length:

The following codes are defined:

Length	Explanation
= 0	The string was not found.
> 0	The string was found at the indicated host presentation space position.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Search Presentation Space** function was successful.
1	Your program is not connected to a host session.
2	An error was made in specifying parameters.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	The search string was not found.

Notes on Using This Function

The following options are supported in a Unicode session for **Search Presentation Space** (6) and function in the same way as in SBCS:

- STRLEN
- SRCHALL
- SRCHFROM

- SRCHFRWD
- SRCHBKWD

Send File (90)

3270	*5250*	*VT*
Yes	Yes	No

The **Send File** function is used to transfer a file from the workstation session where EHLLAPI is running to a host session.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 90.	
Data String	Refer to the examples.	
Length	Length of the target data string. Overridden in EOT mode.	
PS Position	Must be 0.	

Following are examples of the data strings for SBCS

3270 Session

- To send the file to the VM/CMS host system:

```
pc_filename [id:]fn ft [fm] [(option]
```

- To send the file to the MVS/TSO host system:

```
pc_filename [id:]dataset[(member)] [/password] [option]
```

- To send the file to the CICS host system:

```
pc_filename [id:]host_filename [(option]
```

5250 Session

- To send the file to the iSeries, eServer i5, or System i5 host system:

```
pc_filename [id:]library file member [option]
```

Following are examples of the data strings for DBCS:

3270 Session

- To send the file to the VM/CMS host system:

```
pc_filename [id:]fn ft [fm] [(option]
```

- To send the file to the MVS/TSO host system:

```
pc_filename [id:]dataset[(member)] [/password]
[(option]
```

- To send the file to the CICS host system:

```
pc_filename [id:]host_filename [(option]
```

5250 Session

- To send the file to the iSeries, eServer i5, or System i5 host system:

```
pc_filename [id:]library file member [option]
```

Note:

Parameters within [ ] are optional. Available options are listed below. For more information about the options, refer to *Administrator's Guide and Reference*.

Host System	Common Options
VM/CMS	ASCII, JISCII, CRLF, APPEND, LRECL n, RECFM v|f, TIME n, CLEAR, NOCLEAR, PROGRESS, QUIET
MVS/TSO	ASCII, JISCII, CRLF, APPEND, LRECL (n), RECFM (v|f|u), TIME (n), CLEAR, NOCLEAR, PROGRESS, QUIET, BLKSIZE (n), SPACE (n[,m]), AVBLOCK|TRACKS|CYLINDERS
CICS	ASCII, JISCII, CRLF, BINARY, TIME n, CLEAR, NOCLEAR, PROGRESS, QUIET
i5/OS or OS/400	ASCII, JISCII, CRLF, APPEND, SRC, LRECL n, TIME n, CLEAR, NOCLEAR, PROGRESS, QUIET

Note:

JISCII is valid in a DBCS session for Japan only and ASCII is valid for all other SBCS and DBCS sessions.

Note:

Time, if specified, overrides the value in Set Session parameters.

Note:

Other options specified will be passed to the host transfer program. The file transfer program on the host side either uses them, ignores them, or returns an error. Consult the host transfer program documentation to see a complete list of the options supported.

Return Parameters

Return Code	Explanation
2	Parameter error or you have specified a length that is too long (more than 255 bytes) for the EHLLAPI buffer. The file transfer was unsuccessful.
3	File transfer complete.
4	File transfer complete with segmented records.
5	Workstation file name is not valid or not found. File transfer was canceled.
9	A system error was encountered.
27	File transfer terminated because of either a Cancel button or the timeout set by the **Set Session Parameter** (9) function.
101	File transfer was successful (transfer to/from CICS).

If you receive return code 2 or 9, there is a problem with the system or with the way you specified your data string.

Other return codes can also be received which relate to message numbers generated by the host transfer program. For transfers to a CICS host transfer program, subtract 100 from the return code to give you the numeric portion of the message. For example, a return code of 101 would mean that the message number INW0001 was issued by the host. For other host transfer programs, just use the return code as the numerical part of the message. For example, a return of 34 would mean that message TRANS34 was issued by the host transfer program. The documentation for your host transfer program should give more information about the meanings of the specific messages.

Operating system error codes reported by EHLLAPI are greater than 300. To determine the error code, subtract 300 and refer to the operating system documentation for return codes.

Notes on Using This Function

1. Four sets of parameters under the **Set Session Parameters** (9) function are related to this function. They are the QUIET/NOQUIET, STRLEN/STREOT, TIMEOUT=c/TIMEOUT=0, and the EOT=c session options. See items 1 and 2 plus items 7 and 8 for more information.

Send Key (3)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Send Key** function is used to send either a keystroke or a string of keystrokes to the host presentation space.

You define the string of keystrokes to be sent with the calling data string parameter. The keystrokes appear to the target session as though they were entered by the terminal operator. You can also send all attention identifier (AID) keys such as Enter and so on. All host fields that are input protected or are numeric only must

be treated accordingly.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface

Function Number	Must be 3.
Data String	A string of keystrokes, maximum 255. Uppercase and lowercase ASCII characters are represented literally. Function keys and shifted function keys are represented by mnemonics. See Keyboard Mnemonics.
Length	Length of the source data string. Overridden if in EOT mode.
PS Position	NA

Return Parameters

Return Code	Explanation
0	The keystrokes were sent; status is normal.
1	Your program is not connected to a host session.
2	An incorrect parameter was passed to EHLLAPI.
4	The host session was busy; all of the keystrokes could not be sent.
5	Input to the target session was inhibited or rejected; all of the keystrokes could not be sent.
9	A system error was encountered.

Notes on Using This Function

1. The parameters under the **Set Session Parameters** (9) function are related to this function. They are the AUTORESET/NORESET, STRLEN/STREOT, EOT=c, ESC=c, and RETRY/NORETRY session options. See items 1 and 2, 9 and 10, and 19 for more information.
2. Keystrokes cannot be sent to the host session when the keyboard is locked or busy. You can check this condition with the **Wait** (4) function.
3. If the host is busy, input might be rejected.
4. The length of the data string must be explicitly defined by the default length parameter, but it can be defined implicitly by the EOT=c option of the **Set Session Parameters** (9) function.

   When explicitly defining length (see item 1), the value for the length parameter passed by the application must be calculated. For this calculation, allow 2 bytes for compound keystrokes such as `@E` and allow 4 bytes for compound keystrokes such as `@A@C`.

5. To send special control keys, a compound character coding scheme is used. In this coding scheme, one keystroke is represented by a sequence of two to four ASCII characters. The first and third character are always the escape character. The second and fourth character are always a keycode.

   To send the sequence `LOGON ABCDE` followed by the Enter key, you would code the string `LOGON ABCDE@E`. A complete list of these keycodes is represented in Keyboard Mnemonics.

   This compound coding technique allows an ASCII string representation of all necessary keystroke codes without requiring the use of complex hexadecimal key codes.

   The default escape character is `@`. The value of the escape character can be changed to any other character with the ESC=c option of the **Set Session Parameters** (9) function.

6. Users needing higher levels of performance should use the **Copy String to Field** (33) or **Copy String to Presentation Space** (15) function rather than send keystrokes with the **Send Key** (3) function. But remember, only the **Send Key** (3) function can send the special control keys.
7. Refer to **Set Session Parameters (9)** session option 10 (NORESET option) to improve the performance of this function.

   Unless NORESET is required, the reset mnemonic is added to the keystroke strings as a prefix. Therefore, all resettable status except input inhibit are reset.

   The NORESET option is not the same as the **Reset System** (21) function.

8. The keystroke strings, including the AID key, are sent to the host via multiple paths. Each path sends the strings before the first AID key (or including the AID key). EHLLAPI adjusts the string length and the start position of each path. For a host application program, any keystroke might be lost by the AID key process. Therefore, you should not send a keystroke list that includes plural AID keys.
9. During the `@P` (Print) or `@A@T` (Print Presentation Space) process, all requests that update the presentation space are rejected. If the presentation space is busy or the interruption request occurs during the print request, the mnemonic @A@R (Device Reset - Cancel to print the Presentation Space) cancels the request and resets the status.

Keyboard Mnemonics

The keyboard mnemonics provide the ASCII characters representing the special function keys of the keyboard in the workstation. The abbreviation codes make the mnemonics for special keys easy to remember. An alphabetic key code is used for the most common keys. For example, the **Clear** key is *C*, and the **Tab** key is *T*.

Table 7 shows the mnemonics using uppercase alphabetic characters:

Table 7. Mnemonics with Uppercase Alphabetic Characters

Mnemonic	Meaning	3270	5250	VT
@B	Left Tab	Yes	Yes	No
@C	Clear	Yes	Yes	No
@D	Delete	Yes	Yes	No
@E	Enter	Yes	Yes	No
@F	Erase EOF	Yes	Yes	No
@H	Help	No	Yes	No
@I	Insert	Yes	Yes	No
@J	Jump (Set Focus)	Yes	Yes	No
@L	Cursor Left	Yes	Yes	Yes
@N	New Line	Yes	Yes	Yes
@O	Space	Yes	Yes	Yes
@P	Print	Yes	Yes	Yes
@R	Reset	Yes	Yes	No
@T	Right Tab	Yes	Yes	Yes
@U	Cursor Up	Yes	Yes	Yes
@V	Cursor Down	Yes	Yes	Yes
@X*	DBCS (Reserved)	Yes	Yes	No
@Z	Cursor Right	Yes	Yes	Yes

Table 8 shows the mnemonics using a number or lowercase alphabetic characters.

Table 8. Mnemonics with Numbers or Lowercase Characters

Mnemonic	Meaning	3270	5250	VT
@0	Home	Yes	Yes	No
@1	PF1/F1	Yes	Yes	No
@2	PF2/F2	Yes	Yes	No
@3	PF3/F3	Yes	Yes	No
@4	PF4/F4	Yes	Yes	No
@5	PF5/F5	Yes	Yes	No
@6	PF6/F6	Yes	Yes	Yes
@7	PF7/F7	Yes	Yes	Yes

@8	PF8/F8	Yes	Yes	Yes
@9	PF9/F9	Yes	Yes	Yes
@a	PF10/F10	Yes	Yes	Yes
@b	PF11/F11	Yes	Yes	Yes
@c	PF12/F12	Yes	Yes	Yes
@d	PF13	Yes	Yes	Yes
@e	PF14	Yes	Yes	Yes
@f	PF15	Yes	Yes	Yes
@g	PF16	Yes	Yes	Yes
@h	PF17	Yes	Yes	Yes
@i	PF18	Yes	Yes	Yes
@j	PF19	Yes	Yes	Yes
@k	PF20	Yes	Yes	Yes
@l	PF21	Yes	Yes	No
@m	PF22	Yes	Yes	No
@n	PF23	Yes	Yes	No
@o	PF24	Yes	Yes	No
@q	End	Yes	Yes	No
@u	Page Up	No	Yes	No
@v	Page Down	No	Yes	No
@x	PA1	Yes	Yes	No
@y	PA2	Yes	Yes	No
@z	PA3	Yes	Yes	No

Table 9 shows the mnemonics using the combination @A and @alphabetic uppercase (A-Z) key.

Table 9. Mnemonics with @A and @ Uppercase Alphabetic Characters

Mnemonic	Meaning	3270	5250	VT
@A@C	Test	No	Yes	No
@A@D	Word Delete	Yes	Yes	No
@A@E	Field Exit	Yes	Yes	No
@A@F	Erase Input	Yes	Yes	No
@A@H	System Request	Yes	Yes	No
@A@I	Insert Toggle	Yes	Yes	No
@A@J	Cursor Select	Yes	Yes	No
@A@L	Cursor Left Fast	Yes	Yes	No
@A@Q	Attention	Yes	Yes	No
@A@R	Device Cancel (Cancels Print Presentation Space)	Yes	Yes	No
@A@T	Print Presentation Space	Yes	Yes	Yes
@A@U	Cursor Up Fast	Yes	Yes	No
@A@V	Cursor Down Fast	Yes	Yes	No
@A@Z	Cursor Right Fast	Yes	Yes	No

Table 10 shows the mnemonics using the combination @A and @number or @A and @alphabetic lowercase (a-z) key.

Table 10. Mnemonics with @A and @ Lowercase Alphabetic Characters

Mnemonic	Meaning	3270	5250	VT
@A@9	Reverse Video	Yes	Yes	No

@A@b	Underscore	Yes	No	No
@A@c	Reset Reverse Video	Yes	No	No
@A@d	Red	Yes	No	No
@A@e	Pink	Yes	No	No
@A@f	Green	Yes	No	No
@A@g	Yellow	Yes	No	No
@A@h	Blue	Yes	No	No
@A@i	Turquoise	Yes	No	No
@A@j	White	Yes	No	No
@A@l	Reset Host Colors	Yes	No	No
@A@t	Print (Personal Computer)	Yes	Yes	No
@A@y	Forward Word Tab	Yes	Yes	No
@A@z	Backward Word Tab	Yes	Yes	No

Table 11 shows the mnemonics using the combination @A and @special character.

Table 11. Mnemonics with @A and @ Alphanumeric (Special) Characters

Mnemonic	Meaning	3270	5250	VT
@A@-	Field -	No	Yes	No
@A@+	Field +	No	Yes	No
@A@<	Record Backspace	No	Yes	No

Table 12 shows the mnemonics using the combination @S and @alphabetic lowercase.

Table 12. Mnemonics with @S (Shift) and @ Alphabetic Characters

Mnemonic	Meaning	3270	5250	VT
@S@E	Print Presentation Space on Host	No	Yes	No
@S@x	Dup	Yes	Yes	No
@S@y	Field Mark	Yes	Yes	No

DBCS Only: Table 13 shows the mnemonics using the combination @X and @number or @alphabetic lowercase (a-z).

Table 13. Mnemonics Using @X and @Alphabetic Lowercase (For DBCS Only)

Mnemonic	Meaning	3270	5250	VT
@X@1	Display SO/SI	Yes	Yes	No
@X@5	Generate SO/SI	No	Yes	No
@X@6	Display Attribute	No	Yes	No
@X@7	Forward Character	No	Yes	No
@X@c	Split Vertical Bar	No	Yes	No

VT Only: Table 14 shows the mnemonics using the combination @M and @number or @alphabetic lowercase (a-z)

Table 14. Mnemonics Using @M, @Q and @Alphabetic Lowercase (For VT Only)

Mnemonic	Meaning	3270	5250	VT
@M@0	VT Numeric Pad 0	No	No	Yes
@M@1	VT Numeric Pad 1	No	No	Yes

@M@2	VT Numeric Pad 2	No	No	Yes
@M@3	VT Numeric Pad 3	No	No	Yes
@M@4	VT Numeric Pad 4	No	No	Yes
@M@5	VT Numeric Pad 5	No	No	Yes
@M@6	VT Numeric Pad 6	No	No	Yes
@M@7	VT Numeric Pad 7	No	No	Yes
@M@8	VT Numeric Pad 8	No	No	Yes
@M@9	VT Numeric Pad 9	No	No	Yes
@M@-	VT Numeric Pad -	No	No	Yes
@M@,	VT Numeric Pad ,	No	No	Yes
@M@.	VT Numeric Pad .	No	No	Yes
@M@e	VT Numeric Pad Enter	No	No	Yes
@M@f	VT Edit Find	No	No	Yes
@M@i	VT Edit Insert	No	No	Yes
@M@r	VT Edit Remove	No	No	Yes
@M@s	VT Edit Select	No	No	Yes
@M@p	VT Edit Previous Screen	No	No	Yes
@M@n	VT Edit Next Screen	No	No	Yes
@M@a	VT PF1	No	No	Yes
@M@b	VT PF2	No	No	Yes
@M@c	VT PF3	No	No	Yes
@M@d	VT PF4	No	No	Yes
@M@h	VT HOld Screen	No	No	Yes
@M@(space)	Control Code NUL	No	No	Yes
@M@A	Control Code SOH	No	No	Yes
@M@B	Control Code STX	No	No	Yes
@M@C	Control Code ETX	No	No	Yes
@M@D	Control Code EOT	No	No	Yes
@M@E	Control Code ENQ	No	No	Yes
@M@F	Control Code ACK	No	No	Yes
@M@G	Control Code BEL	No	No	Yes
@M@H	Control Code BS	No	No	Yes
@M@I	Control Code HT	No	No	Yes
@M@J	Control Code LF	No	No	Yes
@M@K	Control Code VT	No	No	Yes
@M@L	Control Code FF	No	No	Yes
@M@M	Control Code CR	No	No	Yes
@M@N	Control Code SO	No	No	Yes
@M@O	Control Code SI	No	No	Yes
@M@P	Control Code DLE	No	No	Yes
@M@Q	Control Code DC1	No	No	Yes
@M@R	Control Code DC2	No	No	Yes
@M@S	Control Code DC3	No	No	Yes
@M@T	Control Code DC4	No	No	Yes
@M@U	Control Code NAK	No	No	Yes
@M@V	Control Code SYN	No	No	Yes
@M@W	Control Code ETB	No	No	Yes
@M@X	Control Code CAN	No	No	Yes
@M@Y	Control Code EM	No	No	Yes
@M@Z	Control Code SUB	No	No	Yes
@M@u	Control Code ESC	No	No	Yes
@M@v	Control Code FS	No	No	Yes

@M@w	Control Code GS	No	No	Yes
@M@x	Control Code RS	No	No	Yes
@M@y	Control Code US	No	No	Yes
@M@z	Control Code DEL	No	No	Yes
@Q@A	VT User Defined Key 6	No	No	Yes
@Q@B	VT User Defined Key 7	No	No	Yes
@Q@C	VT User Defined Key 8	No	No	Yes
@Q@D	VT User Defined Key 9	No	No	Yes
@Q@E	VT User Defined Key 10	No	No	Yes
@Q@F	VT User Defined Key 11	No	No	Yes
@Q@G	VT User Defined Key 12	No	No	Yes
@Q@H	VT User Defined Key 13	No	No	Yes
@Q@I	VT User Defined Key 14	No	No	Yes
@Q@J	VT User Defined Key 15	No	No	Yes
@Q@K	VT User Defined Key 16	No	No	Yes
@Q@L	VT User Defined Key 17	No	No	Yes
@Q@M	VT User Defined Key 18	No	No	Yes
@Q@N	VT User Defined Key 19	No	No	Yes
@Q@0	VT User Defined Key 20	No	No	Yes
@Q@a	VT Backtab	No	No	Yes
@Q@r	VT Clear Page	No	No	Yes
@Q@s	VT Edit	No	No	Yes

The following table shows the mnemonics using a special character.

Table 15. Mnemonics with Special Character Keys

Mnemonic	Meaning	3270	5250	VT
@@	@	Yes	Yes	Yes
@$	Alternate Cursor (The Presentation Manager Interface only)	Yes	Yes	Yes
@<	Backspace	Yes	Yes	Yes

The following table shows BIDI key mnemonics:

Table 16. BIDI Key Mnemonics

Mnemonic	Meaning	3270	5250	VT
@:@s	Screen Reverse	Yes	Yes	Yes
@:@n	Bidi Layer	Yes	Yes	Yes
@:@l	Latin Layer	Yes	Yes	Yes
@:@F	Field Reverse	Yes	Yes	No
@:@p	Push	Yes	No	No
@:@e	End Push	Yes	No	No
@:@a	Auto Push	Yes	No	No
@:@r	Auto Reverse	Yes	No	No
@:@d	CSD	Yes	No	No
@:@f	Final	Yes	No	No
@:@i	Isolated	Yes	No	No
@:@m	Middle	Yes	No	No
@:@t	Initial	Yes	No	No

@:@h	Field Shape	Yes	No	No
@:@u	Field Base	Yes	No	No
@:@b	Base	No	Yes	No
@:@o	Close	No	Yes	No
@:@K	Column Heading	No	No	Yes
@:@B	Cursor Direction	No	No	Yes
@:@D	Encoding Mode	No	No	Yes
@:@M	VT Change Display Mode	No	No	Yes (Hebrew only)

The following character keys are interpreted as they are.

a-z	!	'	'	<	}
A-Z	$	(	.	>	[
0-9	%	)	/	=	]
~	&	*	:	?	|
#	"	+	;	{	

1390/1399 Code Page Support

Unicode functionality is supported only on 3270 and 5250 sessions.

STREOT option is not supported in a Unicode session. Please see Set Session Parameters (9) for details.

The session option ESC is not supported in a Unicode session; using this option, you cannot set a Unicode character as an ESC character. Use the default ESC character @ in a Unicode session. Please see Set Session Parameters (9) for details.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 3	
Data String	A Unicode string of keystrokes, maximum 255. Uppercase and lowercase ASCII characters are represented literally. Function keys and shifted function keys are represented by mnemonics. See Keyboard Mnemonics.	
Length	The length of the Unicode string in Unicode characters.	
PS Position	NA	

Return Parameters

Return Code	Explanation
0	The keystrokes were sent; status is normal.
1	Your program is not connected to a host session.
2	An incorrect parameter was passed to EHLLAPI.
4	The host session was busy; all of the keystrokes could not be sent.
5	Input to the target session was inhibited or rejected; all of the keystrokes could not be sent.

9	A system error was encountered.

Notes on Using This Function

Before sending keystrokes to a PCOMM session, be sure that the session is a Unicode session and that the current platform is Windows 2000. .

The string length should indicate the number of Unicode characters and not the number of ANSI characters to be sent.

1137 Code Page Support

Unicode functionality is supported only on 5250 sessions.

STREOT option is not supported in a Unicode session. Please see Set Session Parameters (9) for details.

The session option ESC is not supported in a Unicode session; using this option, you cannot set a Unicode character as an ESC character. Use the default ESC character @ in a Unicode session. Please see Set Session Parameters (9) for details.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 3	
Data String	A Unicode string of keystrokes, maximum 255. Uppercase and lowercase ASCII characters are represented literally. Function keys and shifted function keys are represented by mnemonics. See Keyboard Mnemonics.	
Length	Length of the Unicode data string in bytes. If the length is not a multiple of 2 then an error code of 2 is returned.	
PS Position	NA	

Return Parameters

Return Code	Explanation
0	The keystrokes were sent; status is normal.
1	Your program is not connected to a host session.
2	An incorrect parameter was passed to EHLLAPI.
4	The host session was busy; all of the keystrokes could not be sent.
5	Input to the target session was inhibited or rejected; all of the keystrokes could not be sent.
9	A system error was encountered.

Notes on Using This Function

Before sending keystrokes to a PCOMM session, be sure that the session is a Unicode session. If the session is ANSI and a Unicode string is sent, junk characters will be displayed.

The string length should indicate the number bytes and not the number of Unicode characters to be sent. Therefore the length should be a multiple of 2. If not, a parameter error will be returned by the function.

Set Cursor (40)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Set Cursor** function is used to set the position of the cursor within the host presentation space. Before using the **Set Cursor** function, a workstation application must be connected to the host presentation space.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 40	
Data String	NA	
Length	NA	
PS Position	Desired cursor position in the connected host presentation space	

Return Parameters

Return Code	Explanation
0	Cursor was successfully located at the specified position.
1	Your program is not connected to a host session.
4	The session is busy.
7	A cursor location less than 1 or greater than the size of the connected host presentation space was specified.
9	A system error occurred.

Notes on Using This Function

DBCS Only: If the specified cursor is the second byte of the double-byte character, the cursor moves to the first byte of the character and an error code is not returned.

1137 Code Page Support

The usage of **Set Cursor** in a Unicode session is the same as that for a SBCS session except:

- Unicode functionality is supported only on 5250 sessions.
- In a Unicode session only, if the specified cursor is in the middle of a cluster (for example, a Hindi language cluster), then the cursor is positioned to the beginning of the cluster automatically.

Set Session Parameters (9)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Set Session Parameters** function lets you change certain default session options in EHLLAPI for all sessions. When EHLLAPI is loaded, the default settings for session options are as indicated by the underscored entries in the tables that appear in Session Options . Any, some, or all of these settings can be changed by including the desired option in the calling data string as explained below. Specified settings remain in effect until:

- Changed by a subsequent **Set Session Parameters** (9) function that specifies a new value.
- The **Reset System** (21) function is executed.
- The EHLLAPI application program is terminated.

The following table lists those EHLLAPI functions that are affected by session options. Functions not listed in the table are not affected by any of the session options. Session options that affect each function are indicated by corresponding entries in the "See Items" column. These entries are indexed to the list that follows Call Parameters.

Function Number	Function Name	See Items
1	Connect Presentation Space	11, 23, 24
3	Send Key	1, 2, 9, 10, 19
4	Wait	12
5	Copy Presentation Space	5, 13, 14, 15, 17, 20, 21, 22
6	Search Presentation Space	1, 2, 3, 4
8	Copy Presentation Space to String	5, 13, 14, 15, 17, 20, 21, 22
10	Query Sessions	16, 22
15	Copy String to Presentation Space	1, 2, 13, 14, 18, 20, 21, 22
18	Pause	6
30	Search Field	1, 2, 3, 4, 22
33	Copy String to Field	1, 2, 13, 14, 18, 20, 21, 22
34	Copy Field to String	5, 13, 14, 17, 20, 21, 22
51	Get Key	9, 12
90	Send File	1, 2, 7, 8
91	Receive File	1, 2, 7, 8
101	Connect Window Services	23, 24

Note:

Items 20 and 21 in this table are for DBCS only

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 9.	
Data String	String containing the desired values of those session options that are to be changed. The data string can contain any of the values in the tables of Session Options. The values should be placed on the data string line, separated by commas or blanks. The sets of parameters are explained in terms of the functions they affect.	
Length	Explicit length of the source data string (the `STREOT` option is not allowed).	
PS Position	NA.	

Session Options

The following tables show the session options. The default is underlined.

1. The values in the following table determine how the data string length is defined for functions **Send Key** (3), **Search Presentation Space** (6), **Copy String to Presentation Space** (15), **Search Field** (30), **Copy String to Field** (33), **Send File** (90), and **Receive File** (91).

Value	Explanation
STRLEN	An explicit length is passed for all strings.
STREOT	Lengths are not explicitly coded. Calling (source) data strings are terminated with an EOT character.

2. The statement in the following table is used to specify the character that is used as the end-of-text (EOT) delimiter in the calling (source) data string for EHLLAPI functions **Send Key** (3), **Search Presentation Space** (6), **Copy String to Presentation Space** (15), **Search Field** (30), **Copy String to Field** (33), **Send File** (90), and **Receive File** (91).

Value	Explanation
EOT=c	Allows you to specify the EOT character for string terminators (in STREOT mode). Binary zero is the default. Do not leave a blank after the equal sign.

To be valid, `c` must be entered as a 1-byte string literal character with no preceding blanks. The EOT character specified by this statement is used to determine the length of a calling data string only when the STREOT option (see item 1) is in effect.

3. The values in the following table affect the **Search Presentation Space** (6) and **Search Field** (30) search functions.

Value	Explanation
SRCHALL	The **Search Presentation Space** (6) function and **Search Field** (30) function scan the entire host presentation space or field.
SRCHFROM	The **Search Presentation Space** (6) function and **Search Field** (30) function start from a specified PS position (for `SRCHFRWD`) or end at a specified PS position (for `SRCHBKWD`).

4. The values in the following table affect the **Search Presentation Space** (6) and **Search Field** (30) search functions. They determine the direction for the search.

Value	Explanation
SRCHFRWD	The **Search Presentation Space** (6) function and **Search Field** (30) function perform in an ascending direction.
SRCHBKWD	The **Search Presentation Space** (6) function and **Search Field** (30) function perform in a descending direction. A search is satisfied if the first character of the requested string starts within the bounds specified for the search.

5. The values in the following table determine how attribute bytes are treated for functions **Copy Presentation Space** (5), **Copy Presentation Space to String** (8), and **Copy Field to String** (34).

Value	Explanation
NOATTRB	Convert all unknown values to blanks.
ATTRB	Pass back all codes that do not have an ASCII equivalent as their original values.

NULLATTRB	Convert all field attributes to null characters.

6. The values in the following table affect the **Pause** (18) function.

Value	Explanation
FPAUSE	A full-duration pause lasts for however long you specified in the **Pause** (18) function.
IPAUSE	Interruptible pause. After the **Start Host Notification** (23) function is executed, a host event satisfies a pause.

7. The values in the following table determine whether messages generated by file transfer functions **Send File** (90) and **Receive File** (91) are displayed.

Value	Explanation
NOQUIET	SEND and RECEIVE messages are displayed.
QUIET	SEND and RECEIVE messages are not displayed.

8. The statements in the following table determine how long Personal Communications EHLLAPI waits before it automatically issues a Cancel during execution of file transfer functions **Send File** (90) and **Receive File** (91). To be valid, `c` must be an Arabic number 0-9 or a capital letter J-N and must not be preceded by a blank.

Value	Explanation
TIMEOUT=0	A Cancel is automatically issued following a 20-second (approximate) delay.
TIMEOUT=c	A Cancel is automatically issued following a specified delay. A 1-character indicator from the table below tells Personal Communications how many 30-second cycles it should accept before issuing a Cancel itself. **Character** **Value (in minutes)** **1** 0.5 **2** 1.0 **3** 1.5 **4** 2.0 **5** 2.5 **6** 3.0 **7** 3.5 **8**

	4.0 **9** 4.5 **J** 5.0 **K** 5.5 **L** 6.0 **M** 6.5 **N** 7.0

9. The statement in the following table is used to define the escape character for keystroke mnemonics. This session option affects functions **Send Key** (3) and **Get Key** (51). The value of c must be entered as a 1-byte literal character string with no preceding blanks.

Value	Explanation
ESC=c	Specifies the escape character for keystroke mnemonics (@ is the default). Do not leave a blank after the equal sign. A blank is not a valid escape character.

10. The values in the following table determine whether EHLLAPI automatically precedes strings sent using the **Send Key** (3) function with a reset.

Value	Explanation
AUTORESET	EHLLAPI attempts to reset all inhibited conditions by prefixing all strings of keys sent using the **Send Key** (3) function with a reset.
NORESET	Do not AUTORESET.

11. The values in the following table affect the manner in which the **Connect Presentation Space** (1) command function.

Value	Explanation
CONLOG	Establishes a logical connection between the workstation session and a host session. During Connect, does not jump to the requested presentation space.
CONPHYS	Establishes a physical connection between the workstation session and a host session. During Connect, jumps to the requested presentation space.

12. The values in the following table affect the **Wait** (4) function and **Get Key** (51) function. For each value, there are two different effects, one for each function.

Value	Explanation
TWAIT	For the **Wait** (4) function, waits up to a minute before timing out on XCLOCK (X []) or XSYSTEM. For the **Get Key** (51) function, does not return control to your EHLLAPI application program until it has intercepted a key (normal or AID key based on the option specified under the **Start Keystroke Intercept** (50) function).
LWAIT	For the **Wait** (4) function, waits until XCLOCK (X [])/XSYSTEM clears. This option is not recommended, because control does not return to your application until the host is available.

	For the **Get Key** (51) function, does not return control to your EHLLAPI application program until it has intercepted a key (normal or AID key based on the option specified under the **Start Keystroke Intercept** (50) function).
NWAIT	For the **Wait** (4) function, checks status and returns immediately (no wait). For the **Get Key** (51) function, returns return code 25 (keystrokes not available) in the fourth parameter if nothing is queued matching the option specified under the **Start Keystroke Intercept** (50) function.

Note:

Use of NWAIT is recommended.

13. The values in the following table affect **Copy Presentation Space** (5), **Copy Presentation Space to String** (8), **Copy String to Presentation Space** (15), **Copy String to Field** (33), and **Copy Field to String** (34). Extended attribute bytes (EAB) include extended character attributes and extended field attributes.

Value	Explanation
NOEAB	Pass data only, no EABs.
EAB	Pass the presentation space data with extended attribute bytes. For each character that appears on the screen, 2 bytes of data are passed. Therefore, a buffer twice the size of the presentation space must be preallocated; for example 2 x 1920 = 3840 for a 24-row by 80-column presentation space. Extended attributes for a string of characters may be reported as attributes of the field byte, rather than as attributes of each individual character in the field. In this case, to tell if a particular character or set of characters on a screen is underscored, do a CopyPStoString specifying the position of the field attribute byte (the byte before the field that is displayed on the screen) to get the EAB information that applies to all of the characters in that field.

Note:

When using **EHLLAPI Copy PS to String**, text is copied which should be invisible to the operator. Use the EHLLAPI Set Session Parameters function to set the NODISPLAY option to determine if there is hidden data. This causes EHLLAPI to return nondisplay fields as nulls. Another common procedure for hiding data is to set the foreground and background colors the same (BLACK, for instance) so the text is displayed, but not visible to the human operator. The only way for your application to detect this is to use the EAB and XLATE session parameters and then copying the PS. The foreground/background color of each position is returned and you can determine which characters are invisible.

14. The values in the following table affect **Copy Presentation Space** (5), **Copy Presentation Space to String** (8), **Copy String to Presentation Space** (15), **Copy String to Field** (33), and **Copy Field to String** (34).

Value	Explanation
NOXLATE	EABs are not translated.
XLATE	EABs are translated to the PC color graphics adapter (CGA) format.

15. The values in the following table affect **Copy Presentation Space** (5) and **Copy Presentation Space to String** (8) if NOATTRB and NOEAB are specified.

Value	Explanation
BLANK	Convert all unknown values to X'20'.
NOBLANK	Convert all unknown values to X'00'.

The default value is BLANK. If you want to change the default value to NOBLANK, add the following statement in the PCSWIN.INI file located in the Personal Communications user-class application data directory:

```
[API]
NullToBlank=NO
```

16. The values in the following table affect the presentation space size that is returned by the **Query Sessions** (10).

Value	Explanation
CFGSIZE	Returns the configured size of the connected presentation space. This option ignores any override of the configured size by the host.
NOCFGSIZE	Returns the current size of the connected presentation space.

17. The values in the following table affect **Copy Presentation Space** (5), **Copy Presentation Space to String** (8), and **Copy Field to String** (34).

Value	Explanation
DISPLAY	Copy nondisplay fields in the presentation space to the target buffer area in the same manner as display fields. Current applications function normally.
NODISPLAY	Do not copy nondisplay fields in the presentation space to the target buffer area. Copy the nondisplay fields to the target buffer as a string of null characters. This allows applications to display the copied buffers in the presentation widow without displaying confidential information, such as passwords.

18. The values in the following table affect **Copy String to Presentation Space** (15) and **Copy String to Field** (33).

Value	Explanation
NOPUTEAB	EAB (or EAD for DBCS) is not contained in the data string of **Copy String to Presentation Space** or **Copy String to Field**.
PUTEAB	EAB is contained with character data in the data string of **Copy String to Presentation Space** or **Copy String to Field**.

This option is used for the compatibility with Communication Manager/2. For Communication Manager/2, the data string, which is specified in **Copy String to Presentation Space** or **Copy String to Field**, must be contain EAB (or EAD) with character data when EAB (or EAD) is valid in **Set Session Parameters**. Whereas, for the previous Personal Communications, the data string specified in these functions must consist of character data only even if EAB (or EAD) is valid. But Personal Communications allows that the data string contains EAB (or EAD) by setting PUTEAB to provide the compatibility with Communication Manager/2.

19. The values in the following table affect the **Send Key** (3) function. Keystrokes are not processed if the keyboard is blocked or in use. The options determine whether the function tries to resend the keystrokes until a 4-minute timeout occurs or if the function returns immediately after determining the keyboard is blocked or in use.

Value	Explanation
RETRY	Continues to attempt to send keystrokes until they are sent or until a 4-minute timeout occurs.
NORETRY	Returns immediately after determining the keyboard is blocked or in use.

20. **DBCS Only:** The values in the following table affect **Copy Presentation Space** (5), **Copy Presentation Space to String** (8), **Copy String to Presentation Space** (15), **Copy String to Field** (33), and **Copy Field to String** (34).

Value	Explanation
NOEAD	DBCS attribute characters are not passed.
EAD	Pass the presentation space data and two attribute characters for the double-byte character set (DBCS). (Users receive 2 bytes for each character other than the data. Therefore, a buffer twice the size of the presentation space must be preallocated.)

21. **DBCS Only:** The values in the following table affect **Copy Presentation Space** (5), **Copy Presentation Space to String** (8), **Copy String to Presentation Space** (15), **Copy String to Field** (33), and **Copy Field to String** (34).

Value	Explanation
NOSO	Pass the presentation space data except Shift-in (SI) and Shift-out (SO) control characters.
SO	Pass the presentation space data including translated SI control character to X'0E' and SO control character to X'0F'. The allocated buffer size depends on the length of the stored data.
SPACESO	Pass the presentation space data including translated SI and SO control characters to X'20' (blank). The allocated buffer size depends on the length of the stored data.

22. The values in the following table affect **Copy Presentation Space** (5), **Copy Presentation Space to String** (8), **Copy String to Presentation Space** (15), **Copy String to Field** (33), **Copy Field to String** (34) **Search Field** (30) and **Query Sessions.** (10)

Value	Explanation
EXTEND_PS	5250 emulation supports a presentation space of 24 rows by 80 columns. In some instances, Communication Manager 5250 emulation displays a 25th row. This occurs when either an error message from the host is displayed or when the operator selects the SysReq key. Personal Communications displays 25th row information on row 24, but EHLLAPI normally sees the *real* 24th row. By **EXTEND_PS** option, an EHLLAPI application can use the same interface with Communication Manager EHLLAPI and valid presentation space is extended when this condition occurs.
NOEXTEND_PS	The presentation space is not extended when the above condition occurs. This is the default value.

23. The values in the following table affect the **Connect Presentation Space** (1) and **Connect Window Services** (101) functions. The options specify whether an application can or will share the presentation space to which it is connected with another application. Only one of the following values can be specified with each **Set Session Parameter** call.

Value	Explanation
SUPER_WRITE	The application allows other applications that allow sharing and have write access permissions to concurrently connect to the same presentation space. The originating application performs supervisory-type functions but does not create errors for other applications that share the presentation space.
WRITE_SUPER	The application requires write access and allows only supervisory application to concurrently connect to its presentation space. This is the default value.
WRITE_WRITE	The application requires write access and allows partner or other applications with predictable behavior to share the presentation space.
WRITE_READ	The application requires write access and allows other applications that perform read-only functions to share the presentation space. The application is also allowed to copy the presentation space and perform other read-only operations as usual.
WRITE_NONE	The application has exclusive use of the presentation space. No other applications are allowed to share the presentation space, including supervisory applications. The application is allowed to copy the presentation space and perform read-only operations as usual.
READ_WRITE	The application requires only read access to monitor the presentation space and allows other applications that perform read or write, or both, functions to share the presentation space. The application is also allowed to copy the presentation space and perform other read-only operations as usual.

24. The values in the following table allow applications that have presentation space sharing requirements to limit the sharing to a partner application (an application that was developed to work with it).

Value	Explanation
NOKEY	Allows the application to be compatible with existing applications that do not specify the **KEY** parameter.
KEY$*nnnnnnnn*	Uses a keyword to restrict sharing access to the presentation space that it supports. The keyword must be exactly 8 bytes in length.

Return Parameters

This function returns a length and a return code.

Length:

Number of valid session parameters that are set.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The session parameters have been set.
2	One or more parameters were not valid.
9	A system error was encountered.

1390/1399 and 1137 Code Page Support

Code page 1390/1399 Unicode functionality is available only for 3270 and 5250 sessions. Code page 1137 Unicode functionality is available only for 5250 sessions.

The following session option differences must be noted for 1390/1399 and 1137 code page support in a Unicode session:

- The session option STREOT should not be used for Unicode strings for the following reasons:
  - The session option STREOT specifies that the length of the string is not explicitly given. An EOT character indicates the end of the string. By scanning for the EOT character, the length of the string can be found. This EOT character is stored as a single-byte value. The single-byte EOT character cannot be used for Unicode strings.
    - *Scenario:* A user sets the EOT character as 'A' whose ASCII value is 0X'41'. If the string buffer that the user passes to the function contains a Unicode character, then the low byte of this Unicode character will be taken as the string delimiter. Therefore, a single-byte EOT character cannot be used as a string delimiter.
  - The EOT character cannot be stored as a Unicode character since the **Set Session Parameter** function is independent of the PCOMM session and the same setting applies to all the sessions of PCOMM. If the EOT is to be stored as a Unicode character, then SBCS and DBCS implementations will be affected by the way the EOT character is passed. At present, the EOT character is expected to be a single-byte value.

    Note:

    If you use the session option STREOT, then the results may not be as expected. You can use a single-byte delimiter with the Unicode strings if you are certain that the single-byte delimiter will not be a part of the Unicode values that you are passing in the buffer.
- The session option ESC is not supported in a Unicode session for the same reason as listed for *(EOTREASON)*.
- The session option XLATE is not supported in Unicode. Even if this option is set, it will be ignored.

Start Close Intercept (41)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Start Close Intercept** function allows the application to intercept close requests generated when a user selects the close option from the emulator session window. This function intercepts the close request and discards it until a **Stop Close Intercept** (43) function is requested.

After using this function, your application program can use the **Query Close Intercept** (42) function to determine when a close request has occurred.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

Byte	Definition	
	Standard Interface	Enhanced Interface
Function Number	Must be 41	
Data String	See the following table	
Length	5 or 6	Must be 12
PS Position	NA	

The data string contains the following items.

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID).
	2-4	Reserved.
4-5		The data in these positions is ignored by EHLLAPI. However, no error is caused if the migrating program has data in these positions. This data is accepted to provide compatibility with migrating applications.
6	5	Specify M to request asynchronous message mode (Windows only).
	6-8	Reserved.
2-3	9-12	When M is specified in position 5 (6 for 16-bit), the window handle of the window that receives the message should be set. The message is a return value of RegisterWindowMessage (PCSHLL) (not equal 0).

Return Parameters

This function returns a data string and a return code.

Data String:

If asynchronous message mode is not specified in position 5 (6 for standard interface) and the function is completed successfully, the following data string is returned.

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID).
	2-8	Reserved.
	9-12	4 byte value in which the event object address is returned by EHLLAPI. The application can wait for this event object. (32-bit only).

Data String:

If M (asynchronous message mode) is specified in position 5 (6 for standard interface) and the function is completed successfully, the following data string is returned.

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-8	Reserved
2-3	9-10	Task ID of asynchronous message mode

Note:

If a user selects the close option, an application window receives a message. The message is a return value of RegisterWindowMessage (PCSHLL). The wParam parameter will contain the Task ID returned by this function call. The HIWORD of the IParam parameter will contain the Return Code 26, which shows a close intercept occurred, and the LOWORD of the lParam parameter will contain the function number 41.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Start Close Intercept** function was successful.
1	An incorrect host presentation space was specified.
2	A parameter error occurred.
9	A system error occurred.
10	The function is not supported by the emulation program.

Notes on Using This Function

1. The returned event object or semaphore is in a non-signaled state when the start request function returns. The event object is in the signaled state each time a close request occurs. To receive notification of multiple close request events, put the event object into the signaled state each time using **SetEvent** or the **Query Close Intercept** (42) function.
2. After using this function, your application program can use the **Query Close Intercept** (42) function to determine when a close request has occurred. The application can wait on the returned event object to determine when the event has occurred.
3. This is not an exclusive call. Multiple applications can request this function for the same short session ID.
4. If there are no applications intercepting close requests for a session, any subsequent close requests selected by the user from the emulator operations dialog result in a normal stop requested for that session.

Start Communication Notification (80)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Start Communication Notification** function begins the process by which your EHLLAPI application can determine whether the specified session is connected to a host.

After using this function, the application can use **Query Communication Event** (81) to determine whether the session is connected or disconnected.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Enhanced Interface
Function Number	Must be 80
Data String	Preallocated structure; see the following table
Length	16
PSPosition	NA

The calling data structure contains these elements

Byte	Definition
1	A 1-character presentation space short name (PSID).
2-4	Reserved
5	One of the following values: • The character C asks for notification when the session either disconnects or connects to the host. • The character A requests the asynchronous mode of notification. When A is specified, position 9-12 returns the address of an event object (Windows). The character C must be placed in position 13. • The character M requests the asynchronous message mode of the notification. When M is specified, the event selection character C must be placed in position 13.
6-8	Reserved
9-12	When M is specified in position 5, the window handle of the window that receives the message should be set. The message is a return value of RegisterWindowMessage (PCSHLL)--(not zero).
13	This should contain the character C if position 5 is A or M.
14-16	Reserved

Data String

If A (asynchronous mode) is specified in position 5 of the calling data structure and the function is completed successfully, the following data string is returned:

Byte	Definition
1	A 1-character presentation space short-name (PSID)
2-8	Reserved
9-12	4-byte binary value in which the event object handle is returned by EHLLAPI. The application can wait for this event object.

If M (asynchronous message mode) is specified in position 5 of the calling data structure and the function is completed successfully, the following data string is returned:

Byte	Definition
1	A 1-character presentation space short-name (PSID)
2-8	Reserved
9-10	Task ID of asynchronous message mode

When the session connects or disconnects an application window receives a message. The message is the return value of RegisterWindow Message (PCSHLL). The wParam contains the Task ID returned by the function call. HIWORD of lParam contains a `21` if the session is connected to the host or a 22 if the session is disconnected. The LOWORD of lParam contains the function number `80`.

Return Parameters

Return Code	Definition
0	The function was successful
1	An incorrect PSID was specified
2	An error was made in designating parameters
9	A system error was encountered

Notes on using this Function

1. An application program can issue this function for multiple host sessions. The **Query Communication Event** (81) function can be used to determine the session communication status.
2. If the application chooses the asynchronous option, it can use the Windows SDK call **WaitForSingleObject** to wait until the sessions communication status has

changed.

3. The event object is initially in a non-signaled state. It is signaled each time an event occurs. To receive notification for multiple events the application must put the event object into the non-signaled state each time it is signaled, by using the Windows SDK call **ResetEvent**, or by using function 81 **Query Communications Event**.
4. Multiple calls to this function with the same options from the same application will be ignored.
5. This is not exclusive to one application. Several applications can request this function for the same Session ID.

Start Host Notification (23)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Start Host Notification** function begins the process by which your EHLLAPI application program determines if the host presentation space or OIA have been updated.

After using this function, your application program can use the **Query Host Update** (24) function to determine when a host event has occurred.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 23	
Data String	Preallocated string; see the following table	
Length	6 or 7 implied	16
PS Position	NA	

The calling data string contains these elements:

Byte		**Definition**
Standard	Enhanced	
1	1	One of the following values: • A 1-character presentation space short name (PSID) • A blank or null indicating a request for the host-connected host presentation space
	2-4	Reserved.
2	5	One of the following values: • The character B asking for notification of both host presentation space and OIA updates. • The character O asking for notification of only OIA updates. • The character P asking for notification of only host presentation space updates. • The character A requesting the asynchronous mode of the notification When A is specified, position 9-12 returns the address of an event object. The event selection character B, O, or P must be placed in position 13. • The character M requesting the asynchronous message mode of the notification. When M is specified, the event selection character B, O, or P must be placed in position 13 (7 for 16-bit). • E The character E asking for notification of completion during a printer session.
	6-8	Reserved.
3-4	9-12	When M is specified in position 5 (2 for 16-bit), the window handle of the window that receives the message should be set. The message is a return

		value of RegisterWindowMessage (PCSHLL) (not equal 0).
7	13	One of the following values if position 5 (2 for 16-bit) is A or M: • The character B asking for notification of both host presentation space and OIA updates • The character O asking for notification of only OIA updates • The character P asking for notification of only host presentation update.
	14-16	Reserved.

Return Parameters

This function returns a data string and a return code.

Data String:

If A (asynchronous mode of notification) is specified in position 5 and the function is completed successfully, the following data string is returned:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID).
	2-8	Reserved.
	9-12	4-byte value in which the event object address is returned by EHLLAPI. The application can wait for this event object (32-bit only).

Data String:

If M (asynchronous message mode) is specified in position 5 (2 for standard interface) and the function is completed successfully, the following data string is returned:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-8	Reserved
3-4	9-10	Task ID of asynchronous message mode

Note:

If OIA or presentation space is updated, an application window receives a message. The message is a return value of RegisterWindowMessage (PCSHLL). The wParam parameter contains the Task ID returned by the function call. HIWORD of lParam contains Return Code 21 (shows the OIA is updated), 22 (shows the host presentation space is updated), or 23 (shows both the OIA and the host presentation space are updated), and LOWORD of lParam parameter contains function number 23.

Return Code:

The following codes are defined:

Return Code	Definition
0	The **Start Host Notification** function was successful.
1	An incorrect host presentation space was specified.
2	An error was made in designating parameters.
9	A system error was encountered.

Notes on Using This Function

1. An application program can issue this function for multiple host sessions. The **Pause** (18) function can notify the application when one or more host sessions (PS, OIA, or both of them) are updated. The **Query Host Update** (24) function can be used to determine whether a PS, OIA, or both of them have been updated.
2. If the application chooses the asynchronous option, it can wait for the returned event object or semaphore to determine when a host event has occurred.
3. The event object or semaphore is initially in a non-signaled state and is signaled each time an appropriate event occurs. To receive notification for multiple events, the application must put the event object into the non-signaled state each time it has been signaled using either the **ResetEvent** or the **Query Host Update** (24) function.
4. An application cannot request Start Host Notification more than once with the same options.
5. This is not an exclusive call. Multiple applications can request this function for the same short session ID.

Start Keystroke Intercept (50)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Start Keystroke Intercept** function allows a workstation application to filter any keystrokes sent to a session by a terminal operator. After a call to this function, keystrokes are intercepted and saved until the keystroke queue overflows or until the **Stop Keystroke Intercept** (53) function or **Reset System** (21) function is called. The intercepted keystrokes can be:

- Received through the **Get Key** (51) function and sent to the same or another session with the **Send Key** (3) function
- Accepted or rejected through the **Post Intercept Status** (52) function
- Replaced by other keystrokes with the **Send Key** (3) function
- Used to trigger other processes

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 50	
Data String	See the following table	
Length	Keystroke buffer size EHLLAPI allocates 32 bytes minimum for this buffer.	
PS Position	NA	

The calling data string contains:

Byte		**Definition**
Standard	Enhanced	
1	1	One of the following values: • A specific host presentation space short name (PSID) • A blank or null indicating a request for the host-connected host presentation space
	2-4	Reserved.
2	5	An **option code** character:

		• D for AID keystrokes only. • L for all keystrokes. • M for requesting the asynchronous message mode of the notification (Windows only). When M is specified, a code character D, or L must be placed in position 13 (7 for 16-bit).
	6-8	Reserved.
3-4	9-12	When M is specified in position 5 (2 for 16-bit), the window handle of the window that receives the message should be set. The message is a return value of RegisterWindowMessage (PCSHLL) (not equal 0).
7	13	One of the following values if position 5 (2 for 16-bit) is M: • D for AID keystrokes only. • L for all keystrokes.
	14-16	Reserved.

Data String:

If M (asynchronous message mode) is specified in position 5 (2 for standard interface) and the function is completed successfully, the following data string is returned:

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-8	Reserved
3-4	9-10	Task ID of asynchronous message mode

Note:

If a user sends keystrokes to a session, an application window receives a message. The message is a return value of RegisterWindowMessge (PCSHLL). The wParam parameter contains the Task ID returned by the function call. HIWORD of lParam parameter contains return code 0, which shows that the function was successful, and LOWORD of lParam parameter contains function number 50.

Return Parameters

Return Code	**Explanation**
0	The **Start Keystroke Intercept** function was successful.
1	An incorrect presentation space was specified.
2	An incorrect option was specified.
4	The execution of the function was inhibited because the target presentation space was busy.
9	A system error was encountered. Release is being used.

Notes on Using This Function

1. If a return code of 31 occurs for the **Get Key** (51) function, either:
   - Increase the value of the calling length parameter for this function, or
   - Execute the **Get Key** (51) function more frequently.

   An intercepted keystroke occupies 3 bytes in the buffer. The next intercepted keystroke is placed in the adjacent 3 bytes. When the **Get Key** (51) function retrieves a keystroke (first-in first-out, or FIFO), the 3 bytes that it occupied are made available for another keystroke. By increasing the size of the buffer or the rate at which keystrokes are retrieved from the buffer, you can eliminate buffer overflow.

In the PC/3270, another way to eliminate return code 31 is to operate the PC/3270 emulator in the resume mode.

2. If option code D is provided, EHLLAPI writes intercepted non-AID keys to the presentation space to which they were originally intended, and returns only AID keys to the application.
3. Call the **Stop Keystroke Intercept** (53) function before exiting your EHLLAPI application. Otherwise, keystroke interception remains enabled with unpredictable results.

Start Playing Macro (110)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Start Playing Macro** function invokes a macro. The macro will be executed in the connected session.

Note:

This macro must exist in the Personal Communications user-class application data directory and no extension should be specified in the function call for the macro name.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface
Function Number	Must be 110
Data String	See the following table
Length	Length of macro name, plus 3
PS Position	NA

Byte		Definition
Standard	Enhanced	
1-2		Reserved
3-n		Null terminated macro name

Return Parameters

Return Code	Explanation
0	The **Start Playing Macro** function was successful.
1	The programs is not connected to a host session.
2	An error was made in specifying parameters.
9	A system error was encountered.

Stop Close Intercept (43)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Stop Close Intercept** function allows the application to turn off the **Start Close Intercept** (41) function. After the application has issued the **Stop Close Intercept** function, subsequent close requests result in a normal stop sent to the logical terminal session.

Prerequisite Calls

Start Close Intercept (41)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 43	
Data String	1-character short session ID of the host presentation space	
Length	1	Must be 4
PS Position	NA	

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-4	Reserved

Return Parameters

Return Code	Explanation
0	The **Stop Close Intercept** function was successful.
1	An incorrect host presentation space was specified.
2	An error was made in specifying parameters.
8	No previous **Start Close Intercept** (41) function was issued.
9	A system error occurred.
12	The session stopped.

Stop Communication Notification (82)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Stop Communication Notification** function disables the capability of the **Query Communication Event** (81) function to determine whether any communication events have occurred in the specified Session.

Prerequisite Calls

Start Communication Notification (80)

Call Parameters

	Enhanced Interface
Function Number	Must be 82
Data String	1-character short name of the host presentation space, or a blank or null indicating request for updates to the host-connected presentation space
Length	4 is implied
PSPosition	NA

The calling data structure contains these elements:

Byte	Definition
1	A 1-character presentation space short name (PSID)
2-4	Reserved

Return Parameters

Return Code	Definition
0	The function was successful
1	An incorrect PSID was specified
8	No prior call to **Start Communication Notification** (80) function was called for the PSID
9	A system error was encountered

Stop Host Notification (25)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Stop Host Notification** function disables the capability of the **Query Host Update** (24) function to determine if the host presentation space or OIA has been updated. This function also stops host events from affecting the **Pause** (18) function.

Prerequisite Calls

Start Host Notification (23)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 121	
Data String	See the following note	
Length	1 is implied	Must be 4
PS Position	NA	

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-4	Reserved

Note:

1-character short name of the target presentation space ID, or a blank or a null to indicate a request for the host-connected presentation space.

Return Parameters

Return Code	Definition
0	The **Stop Host Notification** function was successful.
1	An incorrect host presentation space was specified.
8	No previous **Start Host Notification** (23) function was issued.
9	A system error was encountered.

Stop Keystroke Intercept (53)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Stop Keystroke Intercept** function ends your application program's ability to intercept keystrokes.

Prerequisite Calls

Start Keystroke Intercept (50)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 53	
Data String	Short name of the target presentation space (PSID)	
Length	1 is implied	Must be 4
PS Position	NA	

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-4	Reserved

Return Parameters

Return Code	Explanation
0	The **Stop Keystroke Intercept** function was successful.
1	An incorrect presentation space was specified.
8	No prior **Start Keystroke Intercept** (50) function was called for this presentation space.
9	A system error was encountered.

Wait (4)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Wait** function checks the status of the host-connected presentation space. If the session is waiting for a host response (indicated by XCLOCK (X []) or XSYSTEM), the **Wait** function causes EHLLAPI to wait up to 1 minute to see if the condition clears.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 4	
Data String	NA	
Length	NA	
PS Position	NA	

Return Parameters

Return Code	Definition
0	The keyboard is unlocked and ready for input.
1	Your application program is not connected to a valid session.
4	Timeout while still in XCLOCK (X []) or XSYSTEM.
5	The keyboard is locked.
9	A system error was encountered.

Notes on Using This Function

1. The **Wait** function is used to give host requests like those made by the **Send Key** (3) function the time required to be completed.

   Using the **Set Session Parameters** (9) function, you can request the `TWAIT`, `LWAIT`, or the `NWAIT` option. See item 12.

2. You can use this function to see if the host OIA is inhibited.
3. The **Wait** function is satisfied by the host unlocking the keyboard. Therefore, a return code of 0 does not necessarily mean that the transaction has been completed. To verify completion of the transaction, you should use the **Search Field** (30) function or **Search Presentation Space** (6) function combined with the **Wait** function to look for expected keyword prompts.

Window Status (104)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Window Status** function allows the application to query or change a window's presentation space size, location, or visible state.

Prerequisite Calls

Connect Window Services (101)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 104	
Data String	See the following table	
Length	16 or 20	24 or 28
PS Position	NA	

The calling data string can contain:

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2-4	Reserved
2	5	A request option value, select one of the following values: • X'01' for set status **Note:** When the session is embedded In-Place in a compound OLE document, the set form of this function (byte 5 = X'01') always returns 0 but has no effect. • X'02' for query for status • X'03' for query for extended status
	6	Reserved

If the request option value is X'01' (set status):

Byte		**Definition**
Standard	Enhanced	
3-4	7-8	A 16- or 32-bit word containing the status set bits if the request option is 1 (set status). The following codes are valid return values if the request option is set status: **X'0001'** Change the window size. (Not valid with minimize, maximize, restore, or move.) **X'0002'** Move the window. (Not valid with minimize, maximize, size, or restore.) **X'0004'** ZORDER window replacement. **X'0008'**

		Set the window to visible. **X'0010'** Set the window to invisible. **X'0080'** Activate the window. (Sets focus to window and places it in the foreground unless ZORDER is specified. In this case, the ZORDER placement is used.) **X'0100'** Deactivate the window. (Deactivates the window and makes the window the bottom window unless ZORDER is also specified. In this case, the ZORDER placement is used.) **X'0400'** Set the window to minimized. (Not valid with maximize, restore, size, or move.) **X'0800'** Set the window to maximized. (Not valid with minimize, restore, size, or move.) **X'1000'** Restore the window. (Not valid with minimize, maximize, size, or move.)
5-6	9-12	A 16- or 32-bit word containing the X window position coordinate. (Ignored if the move option is not set.)
7-8	13-16	A 16- or 32-bit word containing the Y window position coordinate. (Ignored if the move option is not set.)
9-10	17-20	A 16- or 32-bit word containing the X window size in device units. (Ignored if the size option is not set.)
11-12	21-24	A 16- or 32-bit word containing the Y window size in device units. (Ignored if the size option is not set.)
13-16	25-28	A 16- or 32-bit word containing a window handle for relative window placement. These two words are only for the set option. (Ignored if the ZORDER option is not set.) Valid values are as follows: X'00000003' Place in front of all sibling windows. X'00000004' Place behind all sibling windows.

If the request option value is X'02' (query for status):

Byte		**Definition**
Standard	Enhanced	
3-4	7-8	A 16- or 32-bit word containing X'0000' if the request option is 2 (query for status). The following codes are possible return values if the request option is query for status. More than one state is possible. **X'0008'** The window is visible. **X'0010'** The window is invisible. **X'0080'** The window is activated. **X'0100'** The window is deactivated. **X'0400'** The window is minimized. **X'0800'** The window is maximized.
5-6	9-12	A 16- or 32-bit word containing the X window position coordinate. (Ignored if the move option is not set.)
7-8	13-16	A 16- or 32-bit word containing the Y window position coordinate. (Ignored if the move option is not set.)

9-10	17-20	A 16- or 32-bit word containing the X window size in device units. (Ignored if the size option is not set.)
11-12	21-24	A 16- or 32-bit word containing the Y window size in device units. (Ignored if the size option is not set.)
13-16	25-28	A 16- or 32-bit word containing a window handle for relative window placement. These two words are only for the set option. (Ignored if the ZORDER option is not set.) Valid values are as follows: X'00000003' Place in front of all sibling windows. X'00000004' Place behind all sibling windows.

If the request option value is X'03' (query for extended status):

Byte		**Definition**
Standard	Enhanced	
3-4	7-8	A 16- or 32-bit word containing X'0000' if the request option is 3 (query for extended status). The following codes are possible return values if the request option is query for extended status. More than one state is possible. **X'0008'** The window is visible. **X'0010'** The window is invisible. **X'0080'** The window is activated. **X'0100'** The window is deactivated. **X'0400'** The window is minimized. **X'0800'** The window is maximized.
5-6	9-10	A 16- or 32-bit word containing the current font size in the X-dimension. The value is in screen pels.
7-8	11-12	A 16- or 32-bit word containing the current font size in the Y-dimension. The value is in screen pels.
9-12	13-16	Reserved. This value is always zero.
13-14	17-18	A 16- or 32-bit word containing the row number of the first visible character of the presentation space. This value is usually one, unless the Fixed Size font option is in effect, and the window has been resized such that some of the presentation space is hidden.
15-16	19-20	A 16- or 32-bit word containing the column number of the first visible character of the presentation space.
17-20	21-24	A 16- or 32-bit word containing the presentation space window handle of the session.

Return Parameters

Return Code	Explanation
0	The **Window Status** function was successful.
1	The presentation space was not valid or not connected.
2	An incorrect option was specified.
9	A system error occurred.
12	The session stopped.

Notes on Using This Function

The logical terminal (LT) windows use character cells. When resizing the LT windows, the LT rounds the number to prevent character cell truncation. The requested size and position might be slightly different from what was requested. Follow the set option with a query option to determine the final Presentation Manager window position and size. All x and y coordinate positions and sizes are in pels.

Write Structured Fields (127)

3270	5250	VT
Yes	No	No

The **Write Structured Fields** function allows an application to write structured field data to the host application. If the call specifies S (for Synchronous), the application does not receive control until the **Write Structured Fields** function is completed. If the call specifies A (for Asynchronous), the application receives control immediately after the call. If the call specifies M, the application receives control immediately after the call. The application may wait for the message. In any case (S, A or M), the application provides the buffer address in which data to the host is to be placed.

For a successful asynchronous completion of this function, the following statements apply:

The return code field in the parameter list might not contain the results of the requested I/O. If the return code is not 0, then the request failed. The application must take the appropriate action based on the return code.

If the return code for this request is 0, the application must use the request ID returned with this function call to issue the **Get Request Completion** function call to determine the completion results of the function associated with the request ID. The **Get Request Completion** function call returns the following information:

1. Function request ID
2. Address of the data string from the asynchronous request
3. Length of the data string
4. Return code of the completed function

Prerequisite Calls

Connect for Structured Fields (120) **Allocate Communication Buffer** (123)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 127	
Data String	See the following table	
Length	8, 10, or 14	Must be 20
PS Position	NA	

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID).
	2-4	Reserved.
2	5	S or A or M **S =** Synchronous. Control is not returned to the application until the read is satisfied.

		A = Asynchronous. Control is returned immediately to the application, can wait for the event object. **M =** Asynchronous. Control is returned immediately to the application, can wait for the message.
	6	Reserved.
3-4	7-8	2-byte destination/origin ID.
5-8	9-12	4-byte address of the buffer from which the data is to be written. The buffer must be obtained using the **Allocate Communications Buffer** (123) function.
9-10	13-16	Reserved.
11-12	17-20	When "M" is specified in position 5 (2 for 16-bit), the window handle of the window that receives the message should be set, The message is a return value of RegisterWindowMessage ("PCSHLL") (not equal 0).
13-14		The data in these positions is ignored by EHLLAPI However, no error is caused if the migrating program has data in these positions. This data is accepted to provide compatibility with migrating applications.

Return Parameters

This function returns a data string and a return code.

Data String:

If A (asynchronous) is specified in position 5 (2 for standard interface) and the function is completed successfully, the following data string is returned:

Byte		Definition
9-10	13-14	2-byte Function Request ID. It is used by the **Get Request Completion** (125) function to determine the completion of this function call.
	15-16	Reserved.
	17-20	4-byte value in which the event object address is returned by EHLLAPI. The application can wait for this event object. When the event object is cleared, the application must issue the **Get Request Completion** (125) function call to get results of the **Write Structured Fields** request. (32-bit only).

Note:

An event object is returned for each successful asynchronous request. The event object should not be used again. A new event object is returned for each request and is valid for only the duration of that request.

Data String:

If M (asynchronous message mode) is specified in position 5 (2 for standard interface) and the function is completed successfully, the following data string is returned:

Byte		Definition
9-10	13-14	2-byte Function Request ID. It is used by the **Get Request Completion** (125) function to determine the completion of this function call.
	15-16	Reserved.
11-12	17-18	Task ID of asynchronous message mode.
	19-20	Reserved.

Note:

If the function is completed successfully, an application window receive a message. The message is a return value of RegisterWindowMessage (PCSHLL). The wParam parameter contains the Task ID returned by the function call. HIWORD of lParam parameter contains return code 0, which shows the function was successful, and LOWORD of lParam parameter contains function number 127.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Write Structured Fields** function was successful.
1	A specified host presentation space short session ID was not valid or was not connected.
2	An error was made in specifying parameters.
9	A system error occurred.
11	Resource unavailable (memory unavailable).
34	The message sent inbound to the host was canceled.
35	An outbound transmission from the host was canceled.
36	Request rejected. Lost contact with the host.
37	Failed. The host is inbound disabled.

Notes on Using This Function

1. Return code 35 will be returned when the first **Read Structured Fields** or **Write Structured Fields** is requested after an outbound transmission from the host is canceled. Corrective action is the responsibility of the application.
2. Return code 36 requires that the application disconnect from the emulation program and then reconnect to reestablish communications with the host. Corrective action is the responsibility of the application.
3. Return code 37 will be returned if the host is inbound disabled.
4. The EHLLAPI allows for a maximum of 20 asynchronous requests per application to be outstanding. A return code for unavailable resources (RC=11) is returned if more than 20 asynchronous requests are attempted.
5. If you are using IBM Global Network connections, the maximum number of asynchronous requests is 10.

The structured field data format is as follows:

Offset	Length	Contents
0	1 word	X'0000'
2	1 word	m (message length: the number of bytes of data in the message, the number does not include the buffer header prefix, which contains 8 bytes) This value must be set by the application.
4	1 word	X'0000'
6	1 word	X'0000'
8	8 bytes	Length of the first (or only) structured field message.
10	1 byte	First nonlength byte of the structured field message.
		. . .
m+7	1 byte	Last byte in the structured field message.

Bytes 0 through 7 are the buffer header. These first 8 bytes are used by the emulation program. The user section of the buffer begins with offset 8. Bytes 8 and 9 contain the number of bytes in the first structured field (a structured field message can contain multiple structured fields) including 2 bytes for bytes 8 and 9. Bytes 8 through *m*+7 are used for the structured field message sent to the host.

Synchronous Requests

When **Write Structured Fields** is requested synchronously (the S option in the data string), control is returned to the application only after the request is satisfied. The application can assume:

- The return code is correct.
- The data in the communications buffer (read buffer) is correct.
- The host is no longer processing the **Write Structured Fields** request.

Asynchronous Requests

When **Write Structured Fields** is requested asynchronously (the A option in the data string), the application *cannot* assume:

- The return code is correct.
- The data in the communications buffer (write buffer) is correct.
- The host is no longer processing the **Write Structured Fields** request.

When requested asynchronously, EHLLAPI returns the following values:

- A 16-bit Request ID in positions 13-14 (9-10 for standard interface) of the data string
- The address of a event object in positions 17-20 of the data string.

These are used to complete the asynchronous **Write Structured Fields** call.

The following steps must be completed to determine the outcome of an asynchronous **Write Structured Fields** function call:

- If the EHLLAPI return code is not zero, the request failed. No asynchronous request has been made. The application must take appropriate actions before attempting the call again.
- If the return code is zero, the application should wait until the event object is in the signaled state by using the **Get Request Completion** (125) function. The event object **Get Request Completion** (125) function) and should not be reused. The event object is valid only for the duration of the **Write Structured Fields** function call through the completion of the **Get Request Completion** (125) function call.
- Once the event object is in the signaled state use the returned 16-bit Request ID as the Request ID parameter in a call to the **Get Request Completion** (125) function. The data string returned from the **Get Request Completion** (125) function call contains the final return code of the **Write Structured Fields** function call.

Asynchronous Requests

When **Write Structured Fields** is requested asynchronously (the M option in the data string), the application cannot assume:

- The return code is correct
- The data in the communications buffer (write buffer) is correct
- The host is no longer processing the **Write Structured Fields** request

When requested asynchronously with the M option, EHLLAPI returns the following values:

- A 16-bit request ID in positions 13-14 (9-10 for standard interface) of the data string
- Task ID of asynchronous message mode in position 17-18 (11-12 for standard interface)

These are used to complete the asynchronous **Write Structured Fields** call.

[Top of Page | Previous Page | Next Page | Table of Contents | Index]