PCSHLL Language Reference

(IBM PCOMM 4.01 EHLLAPI)

Purpose

Attachmate publishes this material to assist customers wanting to enable existing or new automation software to work with a legacy application programming interface implemented in a current Attachmate emulator product: WinHLLAPI, EHLLAPI, Attachmate HLLAPI, Enterprise Access Library (EAL), PCSHLL (IBM PCOMM 4.01 EHLLAPI), or WD_API (Wall Data abstraction of HLLAPI).

Attachmate recommends that new automation programs be developed using EXTRA!'s COM (OLE Automation) interfaces. Only when a new automation program requires obscure capability not available in a COM solution should a legacy API be considered. In such situations, Attachmate recommends WinHLLAPI be given first preference, if only because it came about through an industry standardization effort. A second option would be EHLLAPI.

Introduction

An application programming interface, API, is typically provided in a software product to facilitate development of applications that automate tasks employing the software. For tasks that are highly repetitive, time-consuming or error-prone, automation can raise user job satisfaction, reduce operational costs, and improve service to customers.

Personal Communication System High-Level Language (PCSHLL) application programming interface is one such API. Introduced originally by IBM in the middle 1990s for use with Windows 95-based emulators (viz. PC/3270 or PC400), PCSHLL adapted the already-popular programming interface, EHLLAPI. The PCSHLL interface is virtually identical to the older and more widely employed EHLLAPI standard. The main differences involve the format of the data-string parameter used.

Accessing Attachmate 32-bit PCSHLL

In brief, an application accesses this interface by

In brief, an application accesses this interface by

1. ensuring Attachmate software, including dynamic load library PCSHLL32.DLL, is in the system search path, so it will be found and loaded when referenced,

2. ensuring that a session is configured to be associated with a HLLAPI "short-name",

3. declaring in application code specific reference to the PCSHLL entry point and its parameter list. This reference will depend on the application programming language, for example:

C++:

```
extern "C" void FAR PASCAL HLLAPI(LPINT HLLFunc, LPSTR HLLDataStr,
                                      LPINT HLLDatalgth, LPINT PsPos);
```

Visual Basic:

```
Declare Sub HLLAPI Lib "PCSHLL32.DLL"
        (HllFunc%, ByVal HllDataStr$, HllDataLgth%, PsPos%)
```

Header and lib files for EHLLAPI, WinHLLAPI, and Attachmate HLLAPI are distributed with EXTRA!.

PCSHLL Functions

Summary

Function	Number	Purpose
Connect Presentation Space	1	Connects your EHLLAPI program to the specified presentation space. After your program connects to the presentation space, that space becomes the current presentation space and all communication with the host computer occurs through it.
Disconnect Presentation Space	2	Disconnects your PC program from the current presentation space.
Send Key	3	Places a keystroke or string of keystrokes in the current presentation space at the current cursor position.
Wait	4	Tests the status of the current presentation space.
Copy Presentation Space	5	Copies the entire contents of the presentation space to a string in your program. Copied characters are translated into ASCII values before being stored in the data string.
Search Presentation Space	6	Scans the current presentation space for a specified string.
Query Cursor Location	7	Returns the position of the cursor in the current presentation space.
Copy Presentation Space to String	8	Copies the contents of all or part of the current presentation space into a data string defined in your program. Translates copied characters to ASCII before returning the data string.
Set Session Parameters	9	Allows you to change the default session parameters affecting the behavior of various functions.

Function	Number	Purpose
Query Sessions	10	Returns the number of host sessions currently defined. It also returns a 12-byte data string for each session that contains the following information: • Short name of the session • Long name of the session • Type of host session • Size of the presentation space.
Reserve	11	Locks the current presentation space to prevent the terminal operator from entering data.
Release	12	Unlocks the current connected host presentation space, which was locked with Function 11, “Reserve.”
Copy OIA	13	Returns the contents of the Operator Information Area (OIA) for the current presentation space to your program.
Query Field Attribute	14	Returns the attribute of the specified field in the current presentation space.
Copy String to Presentation Space	15	Copies an ASCII string from your program to a specific location in the current presentation space.
Pause	18	Causes your PC program to wait a specific amount of time for an event to occur. Use this function instead of a timing loop.
Query System	20	Returns a 35-byte data string indicating the support level provided to your program by the underlying low-level and high-level RAM-resident modules (and other system related values).
Reset System	21	Reinitializes EHLLAPI to the default options in Function 9, “Set Session Parameters.” Also stops host event notification, releases any reserved host sessions, and disconnects any connected host presentation sessions. Must be used at conclusion of HLLAPI work with a session.
Query Session Status	22	Returns an 18-byte data string with information about a session.
Start Host Notification	23	Allows your EHLLAPI program to determine if the presentation space or OIA has been updated.

Function	Number	Purpose
Query Host Update	24	In conjunction with Function 23, “Start Host Notification,” enables your EHLLAPI program to determine if the host presentation space or OIA has been updated since the last time this request was made.
Stop Host Notification	25	Deactivates host notification and disables Function 24, “Query Host Update.” Also prohibits host events in the specified host session from affecting Function 18, “Pause.”
Search Field	30	Searches the designated field within the current presentation space for the occurrence of a specified string.
Find Field Position	31	Returns the beginning position of a target field in the currently connected presentation space.
Find Field Length	32	Returns the length of a target field in the current presentation space.
Copy String to Field	33	Copies a string of characters from your program to a specified field in the current presentation space.
Copy Field to String	34	Copies all the characters from a specified field in the current presentation space to a data string in your program. Translates copied characters to ASCII before returning the data string.
Set Cursor	40	Allows you to position the cursor within the current presentation space.
Start Close Intercept	41	Intercepts all close requests and suppresses them until your program calls Function 43, “Stop Close Intercept.”
Query Close Intercept	42	Lets your PC program determine whether an attempt has been made to close an emulator window.
Stop Close Intercept	43	Lets your PC program turn off Function 41, “Start Close Intercept.”
Start Keystroke Intercept	50	Lets your PC program read and evaluate keystrokes entered by a terminal operator.

Function	Number	Purpose
Get Key	51	Lets your PC program retrieve keystrokes from a session and accept, process, or reject the keystrokes.
Post Intercept Status	52	Once this function is called, the PC beeps when a keystroke has been rejected after calling Function 51, “Get Key.”
Stop Keystroke Intercept	53	Disables your PC program's ability to intercept keystrokes.
Send File	90	Sends a file to a host. Allows you to embed the appropriate file transfer SEND command within your EHLLAPI program.
Receive File	91	Receives a file from a host. Lets you embed the appropriate file transfer receive command in your EHLLAPI program.
Convert Position or RowCol	99	Performs one of the following functions, depending on the requesting parameters passed by your program: • Converts the presentation space position into row and column coordinates. • Converts row and column coordinates into a presentation space position.
Connect Window Services	101	Lets your PC program manage the presentation space windows.
Disconnect Window Services	102	Breaks the Window Services connection between your PC program and the specified host presentation space.
Query Window Coordinates	103	Allows your PC program to request the window coordinates of a presentation space.
Window Status	104	Lets your PC program query or change a window's presentation space size, location, or visible state.
Change Switch List LT Name	105	Lets your PC program change or reset a switch list for a selected logical terminal.
Change PS Window Name	106	Lets your PC program define a new name for the presentation space window or redefine the window to the default name.

What information is provided for each function?

For each PCSHLL function, the following information is presented:

- The function number together with its formal name,
- Brief description of the function purpose,
- Prerequisites
- Applicable session parameters
- Call parameters
- Return parameters
- Notes or tips

Prerequisites

Many PCSHLL functions require another function to be called and successfully completed before the desired call is issued. If the prerequisites are not satisfied, an error code is returned. If *None* appears, no prerequisite calls are necessary.

Applicable session parameters

Function 9, “Set Session Parameters,” allows an application program to set optional PCSHLL features, or session parameters. This section indicates whether any session parameters affect this function and, if so lists the applicable parameters and how they affect the function. If the function is not affected by any session parameters, *None* appears.

Call parameters

Function number

In this parameter, you specify the function number to be called.

Data string

This parameter could be a string of characters or a string of concatenated data items, with enough space set aside to receive the requested output. If the calling data string has special requirements, they will be discussed in “Data string features.”

Data length

Use an unsigned integer to give the length of either a character string or a list of data items. Use an End of Text (EOT) character at the end of each string that is sent to PCSHLL if you do not want to calculate your string length. If you like, you can change the established EOT character through Function 9, “Set Session Parameters.”

Presentation space position

If the presentation space (PS) position parameter is required, it should be an integer representing a position within the EXTRA! host session.

The chart below shows, for each 3270 model number, the range of values that may be specified for PS position.

Model number	Range of PS position values
2	1–1920
3	1–2560
4	1–3440
5	1–3564

In this manual, the words “Not applicable” may appear next to some parameters. While it may appear as if these parameters are not required, they still must be present in an application program before it can call a function. Call parameters must be properly declared, then listed in a call statement. Syntax of the call statement will vary, depending on the programming language.

Return parameters

Parameters returned to an application program by the functions are explained in this section. These parameters include the data string, data length, and result code (PS position).

Data string

If the data string parameter is returned, it will either be a string of characters or a string of concatenated data items. If the returning data string has special features, they will be discussed in “Data string features.”

Data length

When returned, the data length parameter either gives you the length of the data string or it provides the position of the PS.

Result code (PS position)

When a function call returns, the result code takes the place of the PS position call parameter. This code tells whether the function was successful or it encountered a problem. Each function has a result code table that can be used to translate the code into its message. All functions pass a result code in the fourth parameter. Many functions use standard result codes (zero means the function completed successfully, 9 means a system error was encountered, and so on). However, certain functions use slightly different interpretations of the result codes. See the function descriptions in this chapter for details on result codes for each function.

Notes or tips

This area presents guidelines and tips on how to use the function in an application program, along with technical information about the function.

Function 1: Connect Presentation Space

This function connects a PCSHLL application to a specified presentation space (PS). If the application already has a connection, the connected PS is automatically disconnected, and a new connection established. An exclusive connection is established with PCSHLL between the client application program and the PS that requires the target session to be defined in the current EXTRA! configuration. An application program must call this function before requesting any of the following-listed functions.

Number	**Name**
2	Disconnect Presentation Space
3	Send Key
4	Wait
5	Copy Presentation Space
6	Search Presentation Space
7	Query Cursor Location
8	Copy Presentation Space to String
11	Reserve
12	Release
13	Copy OIA
14	Query Field Attribute
15	Copy String to Presentation Space
30	Search Field
31	Find Field Position
32	Find Field Length
33	Copy String to Field
34	Copy Field to String
40	Set Cursor

Prerequisites

Target sessions must be defined in the current EXTRA! configuration.

Applicable session parameters

The following session parameters from Function 9 affect this function.

WRITE_SUPER (default)
This application requires write access and allows only supervisory applications to connect to its PS.

WRITE_WRITE
This application requires write access and allows other applications that have predictable behavior to connect to its PS.

WRITE_READ
This application requires write access and allows other applications to use read-only functions on its PS.

WRITE_NONE
This application requires exclusive access to its PS. No other applications may access its PS.

SUPER_WRITE
This supervisory application allows applications with write access to share the connected PS. The application program setting this parameter will not cause errors for other applications but will provide only supervisory-type functions.

WRITE_READ
This application requires read-only access and allows other applications that perform read-only functions to connect to its PS.

CONLOG (default)
When Function 1, "Connect Presentation Space," is called, the emulator session corresponding to the target PS does **not** become the active application. The calling application remains active. Likewise, when Function 2, "Disconnect Presentation Space," is called, the calling application remains active.

CONPHYS
Calling Function 1, "Connect Presentation Space," makes the emulator session corresponding to the target PS the active application (does a physical connect). Note that this parameter is honored only when there is host access software attached to the session. During Function 2, "Disconnect Presentation Space," the host access software becomes the active application.

Call parameters

An application program must pass the following parameters when calling this function:

Function	1
Data string	A four-byte string, the first character being 1-character PS short name; which must be a letter of the alphabet (A–Z).
Data length	4
PS position	Reserved.

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	**Description**
0	The function was successful; the host presentation space is unlocked and ready for input.
1	An invalid host presentation space ID was entered.
4	Connection succeeded, but the host PS was busy.
5	Connection succeeded, but the host PS was locked (input inhibited).
9	A system error occurred.
11	The requested PS was in use by another application.

Example

```
int HllFunc = 1;
char HllDatStr[1];
/* Short name of session to connect */
HllDataStr[0] = 'B';
int HllDataLgth;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Hint

If the EXTRA! session specified has not already been started when this function is called, calling this function will start the session in hidden state. Because function 1 returns immediately, the result code will be 5 (PS locked). Before attempting to use the session, the application should repeatedly call function 4, "Wait," until a 0 (Success) result code is obtained.

Function 2: Disconnect Presentation Space

This function disconnects an application from its currently connected PS and releases any PS keyboard reservation, but does not reset session parameters to defaults. After calling this function, the application cannot call functions that depend on connection to a PS.

An application automatically disconnects from the currently connected PS when it connects to another PS.

A PCSHLL application program should call this function to disconnect from the currently connected PS before exiting.

Prerequisites

Function 1, "Connect Presentation Space."

Applicable session parameters

The following session parameter from Function 9 affects this function.

CONPHYS
If set (as opposed to default CONLOG), the calling application becomes activated when PCSHLL function 2 is called.

Call parameters

An application program must pass the following parameters when calling this function:

Function	2
Data string	Not applicable.
Data length	Not applicable.
PS position	Reserved.

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
1	The application was not connected with a host PS.
9	A system error occurred.

Example

```
int HllFunc = 2;
char HllDatStr[1];
int HllDataLgth;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Hint

This function only logically disconnects an application from an EXTRA! session. It does not signal the end of PCSHLL interaction by the application. In contrast, a call to function 21, “Reset System,” frees resources used by EXTRA! and allows disconnected session(s) to close when the application exits.

Function 3: Send Key

This function sends a string of up to 255 keystrokes to the currently connected PS. The session cannot receive keystrokes unless the keyboard is unlocked. After the first AID key is processed by the function, keystrokes are no longer accepted and the rest of the string is ignored.

It is possible to represent all necessary keystrokes, including special function keys in ASCII, by using an escape character (the default value is @) followed by the appropriate key code. **Appendix B, "Keyboard Mnemonics,"** provides a complete list of these key codes.

PCSHLL changes the cursor position to the position immediately following the entered string.

Prerequisites

Function 1, "Connect Presentation Space."

The keyboard must be unlocked before keystrokes will be accepted.

Applicable session parameters

The following session parameters from Function 9 affect this function.

STRLEN (default)
String parameters are passed with an explicit length (specified in Data length).

STREOT
String parameters are passed with the character specified in the EOT session parameter denoting the string end.

EOT= char
This character denotes the end of a string when the STREOT session parameter has been set. Null (/0) is the default value.

ESC= char
Specifies the escape character for keystroke mnemonics ("@" is the default). Blank is not a valid escape value.

AUTORESET (default)
Attempts to reset inhibited conditions by adding the RESET prefix to all keystroke strings sent.
"
NORESET
Does not add RESET prefix to key strings.

Call parameters

An application program must pass the following parameters when calling this function:

Function	3
Data string	A string of maximum 255 characters (keystrokes)to be sent to the host PS. The string must end with an EOT character if STREOT is set in Function 9.
Data length	The string length. Overridden if STREOT is set in Function 9.
PS position	Reserved.

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
1	The application was not connected with a host PS.
2	An incorrect parameter was entered.
4	Host session was busy; not all keystrokes were sent.
5	Host session was inhibited, not all keystrokes were sent.
9	A system error occurred.

Example

```
int HllFunc = 3;
char HllDataStr[10];
/* Send "Hello" followed by Enter keystroke */
strcpy (HllDataStr, "Hello@E");
/* Length of data including Escape character */
int HllDataLgth = 7;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Tips

- For increased performance, an application may send entire strings using Function 33, “Copy String to Field,” or Function 15, “Copy String to Presentation Space,” rather than using this function; however, only function 3 may send special control keys.
- If the keystroke string is longer than 255 characters (which is the Send Key function’s limit), use multiple calls to the Send Key function.

Function 4: Wait

This function provides current status of XCLOCK or XSYSTEM conditions of the OIA. (Function 9, "Set Session Parameters," allows a program to vary the amount of time this function will wait for the OIA to clear.)

The Wait function is not a good method for determining when the host is ready for input. This function is provided to determine if the terminal session can accept keystrokes (using "Send Key" or a copy function). To determine when the host is ready, the application should search the screen for key fields, usually near the bottom of the screen. Another method is to query the cursor position until it is located at the correct field. Because host applications are so different and a terminal cannot determine when a host application is ready for input, the PCSHLL application should determine when the host is ready for more input.

If the application program is already in a Wait, Pause, Get Key, or synchronous file transfer, the request for another delay is rejected.

Prerequisites

Function 1, "Connect Presentation Space."

Applicable session parameters

The following session parameters from Function 9 affect this function.

TWAIT (default)
The function waits up to one minute before it times out on XCLOCK or XSYSTEM.

LWAIT
The function waits until XCLOCK or XSYSTEM clears, then returns control to the application once the host becomes available.

NWAIT
The function does not wait but returns immediately with XCLOCK and XSYSTEM status.

Call parameters

An application program must pass the following parameters when calling this function:

Function	4
Data string	Not applicable.
Data length	Not applicable.
PS position	Reserved.

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful; host PS is unlocked and ready for input.
1	The application was not connected with a host PS.
4	Function timed out while in XCLOCK or XSTATUS state.
5	Keyboard is locked.
9	A system error occurred.

Example

```
int HllFunc = 4;
char HllDatStr[1];
int HllDataLgth;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Tips

- This function can be used together with a function like Function 6, "Search Presentation Space," to determine when the host is ready for the next input.
- The PCSHLL application should consider relative machine speed. For example, a host may complete its task during a Wait on a slow machine, but a faster machine may need another approach, as noted earlier.

Function 5: Copy Presentation Space

This function copies the currently connected PS to a string allocated in the calling application.

Prerequisites

Function 1, "Connect Presentation Space."

Applicable session parameters

The following session parameters from Function 9 affect this function.

NOATTRB (default)
Attribute bytes and other characters not displayable in ASCII are translated into blanks.

ATTRB
Attribute bytes and other characters not displayable in ASCII are not translated.

EAB
Extended Attribute Bytes (EABs) are copied. Two characters are placed in the application data string for each one that appears in the PS. The EAB is the second character. To accommodate this, the application program must allocate a data string that is twice the number of displayable characters to be copied from the presentation space of the current display model.

NOEAB (default)
EABs are not copied.

XLATE
EABs are translated to CGA text mode attributes.

NOXLATE (default)
EABs are not translated.

DISPLAY (default)
Non-display fields are copied to the target buffer in the same manner as the display fields.

NODISPLAY
Non-display fields are copied to the target buffer as a string of nulls. This allows an application program to display the copied buffer in the presentation window without displaying confidential information, such as passwords.

Call parameters

An application program must pass the following parameters when calling this function:

Function	5
Data string	A string large enough to accommodate data from the current PS display Model (including EABs if requested). See chart below.
Data length	Not applicable (PS length implied).
PS position	Reserved.

Model number	Data string length required
2	1920 (3840 with EABs)
3	2560 (5120 with EABs)
4	3440 (6880 with EABs)
5	3564 (7128 with EABs)

Return parameters

Data string
Function replaces content of call parameter Data string with text from the presentation space.

Refer to Appendix D, "Extended Attributes," for information on EAB interpretation.

Result code
Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	Success; text from the PS has been copied to data string.
1	The application was not connected with a host PS.
4	The copy was successful, but PS was waiting for host response.
5	The copy was successful, but the keyboard is locked.
9	A system error occurred.

Example
```
int HllFunc = 5;
/* Reserve string for text from Model 2 screen w/o EABs */
char HllDataStr[1920];
int HllDataLgth;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Tips
- Use this function only when the entire PS is needed; otherwise, use Function 8, "Copy Presentation Space to String," or Function 34, "Copy Field to String."
- Use Function 10, "Query Sessions," or Function 22, "Query Session Status," to check host session PS size (which may be changed by the host).

Function 6: Search Presentation Space

This function searches the currently connected PS for first or last occurrence of specified text.

This function is useful for determining whether a specific host panel is present. For example, if the application is expecting a prompt before sending data, this function will search for the message or string before moving on. If the prompt or message is not found, the application program can call Function 18, "Pause," or Function 24, "Query Host Update," and continue to call Function 6 until the string is found.

Prerequisites

Function 1, "Connect Presentation Space."

Applicable session parameters

The following session parameters from Function 9 affect this function.

SRCHALL and SRCHFRWD (default)
The function scans the entire PS for the first occurrence of the specified string.

SRCHALL and SRCHBKWD
The function scans the entire PS for the last occurrence of the specified string.

SRCHFROM and SRCHFRWD
The function scans the PS from the specified PS position for the first occurrence of the string.

SRCHFROM and SRCHBKWD
The function scans the PS from the specified PS position for the last occurrence of the string.

STREOT
If set, a string must end with an EOT character

Call parameters

An application program must pass the following parameters when calling this function:

Function	6
Data string	Text to be searched for in the PS
Data length	Length of the data string. (Ignored if in EOT mode.)
PS position	Start position where the search function is to begin (SRCHFRWD) or to end (SRCHBKWD). This parameter is ignored if SRCHALL is set.

Return parameters

PS Position

Function replaces the value of call parameter Data length with the PS position where specified text was found, or 0 if the text was not found.

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful (the specified text was found).
1	The application was not connected with a host PS.
2	An incorrect parameter was entered.
7	An invalid PS position was specified for beginning the search
9	A system error occurred
24	The specified text was not found.

Example

```
int HllFunc = 6;
char HllDataStr[10];
/* Text to search for: "Hello" */
strcpy (HllDataStr, "Hello");
int HllDataLgth = 5;
/* Start search at PS position 199 */
int PsPos = 199;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Tips

- The SRCHFROM option is useful when you are searching for a string that may occur several times.
- The search carried out by this function is case-sensitive.
- To determine when the host is ready for input, the application should search the screen for key fields, usually near the bottom of the screen.

Function 7: Query Cursor Location

This function returns the position of the cursor in the currently connected PS.

Prerequisites

Function 1, “Connect Presentation Space.”

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	7
Data string	Not applicable.
Data length	Not applicable.
PS position	Reserved.

Return parameters

PS Position

Function replaces the value of call parameter Data length with the PS position of the cursor.

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	**Description**
0	The function was successful (the specified text was found).
1	The application was not connected with a host PS.
9	A system error occurred

Example

```
int HllFunc = 7;
char HllDataStr[];
int HllDataLgth;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Tips

- This function is one method of determining whether a host session is at a particular screen, assuming the position where the cursor will appear on that screen is known in advance.
- To make this determination, the application can repeatedly query cursor position until it is located at the correct field.

Function 8: Copy Presentation Space to String

This function copies all or part of the currently connected PS to a string allocated in the calling application.

Prerequisites

Function 1, "Connect Presentation Space."

Applicable session parameters

The following session parameters from Function 9 affect this function.

NOATTRB (default)
Attribute bytes and other characters not displayable in ASCII are translated into blanks.

ATTRB Attribute bytes and other characters not displayable in ASCII are not translated.

EAB
Extended Attribute Bytes are copied. Two characters are placed in the application data string for each one that appears in the PS. The EAB is the second character. To accommodate this, the application program must allocate a data string that is twice the number of displayable characters to be copied. For example, 160 bytes should be allotted to copy the first 80 characters with EABs.

NOEAB (default)
Extended Attribute Bytes are not copied.

XLATE
Extended Attribute Bytes are translated to CGA text mode attributes.

NOXLATE (default)
Extended Attribute Bytes are not translated.

DISPLAY (default)
Text in non-display fields is copied to the data string in the same manner as display fields.

NODISPLAY
Text in non-display fields is copied to the data string as null characters.

Call parameters

An application program must pass the following parameters when calling this function:

Function	8
Data string	A string of sufficient size to hold data requested from the PS, including EABs if requested
Data length	The number of characters allocated in Data string.
PS position	The PS position where the copying should begin.

Return parameters

Data string

Function replaces content of call parameter Data string with text from the presentation space.

Refer to Appendix D, “Extended Attributes,” for information on EAB interpretation.

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful; requested data was copied to the string.
1	The application program was not connected to a valid PS.
2	String length was specified as zero, or extended past the end of the PS.
4	Requested data was copied, but the PS was waiting for host response.
5	Requested data was copied, but the keyboard was locked.
7	An invalid PS position was specified for beginning the copy.
9	A system error occurred.

Example

```
int HllFunc = 8;
/* At least the size of returned data */
char HllDataStr[5];
/* Length of string to copy */
int HllDataLgth = 5;
/* Start position to copy */
int PsPos = 199;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 9: Set Session Parameters

This function sets session parameters in PCSHLL. Parameters set with this function affect many other PCSHLL functions, as noted in individual function descriptions (“Applicable session parameters”) and in the descriptions of this function’s call parameters.

Session parameter values set using this function remain in effect until one of the following occurs:

- Function 21, “Reset System,” which resets the session parameters to default values
- A new value is specified by a second function 9 call
- The PCSHLL client application program terminates

Prerequisites

None.

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	9
Data string	A string containing one or more session parameters, which can be separated by commas or blanks. The following sections explain the possible session parameters and values.
Data length	The number of characters in Data string. (EOT is not allowed.)
PS position	Reserved.

Copy parameters

The following session parameters affect all copy functions.

ATTRB
EBCDIC characters that cannot be translated to displayable ASCII characters are not translated.

NOATTRB (default)
EBCDIC characters that cannot be translated to displayable ASCII characters are translated to blanks (0x20).

NULLATTRB
Convert field attributes to null characters (0x00).

EAB
Extended Attribute Bytes are copied along with data.

NOEAB (default)
EABs are not copied (data only).

STRLEN (default)
String parameters are passed with an explicit length (specified in Data length).

STREOT
String parameters are passed with the character specified in the EOT session parameter denoting the string end.

EOT= char
This character denotes the end of a string when the STREOT session parameter has been set. Null (/0) is the default value.

XLATE
Copied Extended Attribute Bytes are translated to CGA color codes.

NOXLATE (default)
Copied Extended Attribute Bytes are returned without translation.

DISPLAY (default)
Non-display fields are copied to the target buffer in the same manner as the display fields.

NODISPLAY
Non-display fields are copied to the target buffer as nulls.

BLANK (default)
Null characters are converted to spaces (returned as X'20').

NOBLANK
Null characters are not converted (returned as X'00').

Connect parameters

The following session parameters affect Function 1, “Connect Presentation Space,” and Function 2, “Disconnect Presentation Space.”

CONLOG (default)
When Function 1, “Connect Presentation Space,” is called, the emulator session corresponding to the target PS does **not** become the active application. The calling application remains active. Likewise, when Function 2, “Disconnect Presentation Space,” is called, the calling application remains active.

CONPHYS
Calling Function 1, “Connect Presentation Space,” makes the emulator session corresponding to the target PS the active application (does a physical connect). Note that this parameter is honored only when there is host access software attached to the session. During Function 2, “Disconnect Presentation Space,” the host access software becomes the active application.

WRITE_SUPER (default)
This parameter is set by a client application program that requires write access and allows only supervisory applications to connect to its PS.

WRITE_WRITE
This parameter is set by a client application program that requires write access and allows other applications that have predictable behavior to connect to its PS.

WRITE_READ
This parameter is set by a client application program that requires write access and allows other applications to use read-only functions on its PS.

WRITE_NONE
This parameter is set by a client application program that requires exclusive access to its PS. No other applications will have access to its PS.

SUPER_WRITE
This parameter is set by a supervisory client application program that allows applications with write access to share the connected PS. The client application program setting this parameter will not cause errors for other applications, but will provide only supervisory-type functions.

WRITE_READ
This parameter is set by a client application program that requires read-only access and allows other applications that perform read-only functions to connect to its PS.

KEY$nnnnnnnn
This parameter allows the client application program to restrict sharing the PS. The keyword must be exactly 8 bytes long.

NOKEY (default)
This parameter allows the client application program to be compatible with existing applications that do not specify the KEY parameter.

Esc/Reset parameters

The following session parameters affect Function 3, “Send Key,” and Function 51,“Get Key.”

ESC= char
Specifies the escape character for keystroke mnemonics (“@” is the default). Blank is not a valid escape value.

AUTORESET (default)
Attempts to reset all inhibited conditions by adding the prefix RESET to all keystroke strings sent using Function 3, “Send Key.
”

NORESET
Does not add RESET prefix to function 3 key strings.

Search parameters

The following session parameters affect all search functions.

SRCHALL (default)
Scans the entire PS or field.

SRCHFROM
Starts the scan from a specified location in the PS or field.

SCRCHFRWD (default)
Performs the scan in an ascending direction.

SRCHBKWD
Performs the scan in a descending direction through the PS or field.

Wait parameters

The following session parameters affect Function 4, “Wait,” and Function 51, “Get Key.”

TWAIT (default)
For Function 4, “Wait,” TWAIT waits up to a minute before timing out on XCLOCK or XSYSTEM.
For Function 51, “Get Key,” TWAIT does not return control to the PCSHLL client application program until it has intercepted a key (a normal or AID key, based on the option code specified under Function 50, “Start Keystroke Intercept”).

LWAIT
For Function 4, “Wait,” LWAIT waits until XCLOCK / XSYSTEM clears. This option is not recommended because XSYSTEM or permanent XCLOCK will prevent control being returned to the application.
For Function 51, “Get Key,” LWAIT does not return control to your application until it has intercepted a key. The intercepted key could be a normal or AID key, based on the option specified under Function 50, “Start Keystroke Intercept.”

NWAIT
For Function 4, “Wait,” NWAIT checks status and returns immediately (no wait).
For Function 51, “Get Key,” NWAIT returns code 25 (keystroke not available) if nothing matching the option specified under Function 50, “Start Keystroke Intercept,” is queued.

Pause parameters

The following session parameters affect Function 18, “Pause,” determining the type of pause to perform.

Note An application can make multiple Function 23 calls, and an event satisfying any of the calls will interrupt the pause.

FPAUSE (default)
Full-duration pause. Control returns to the calling application when the number of half-second intervals specified in the Function 18 call have elapsed.

IPAUSE
Interruptible pause; Control returns to the calling application when a system even specified in a preceding Function 23, “Start Host Notification,” call has occurred, or the number of half-second intervals specified in the Function 18 call have elapsed.

Time parameters

The following session parameters affect Function 90, “Send File,” and Function 91, “Receive File.”

NOQUIET (default)
Displays SEND and RECEIVE messages showing progress of the file transfer.

QUIET
Does not display SEND and RECEIVE messages.

TIMEOUT=char
Specifies how much time shall elapse before CTRL BREAK is issued to terminate an in-progress file transfer. (Blank is not accepted.)

Character	**Seconds**
0	20
1	30
2	60
3	90
4	120
5	150
6	180
7	210
8	240
9	270
J	300
K	330
L	360
M	390
N	420

OIA parameters

The following session parameters affect Function 13, “Copy OIA,” and specify the format for the data returned by the function.

OLDOIA (default)
OIA data are returned in EBCDIC. Since OIA data are always returned in ASCII format in 5250 support, OLDOIA is accepted but ignored.

NEWOIA
OIA data are returned in ASCII format.

PS size parameters

The following session parameters affect Function 10, “Query Sessions.”

NOCFGSIZE
Function 10 returns the current size of the connected PS.

CFGSIZE (default)
Function 10 ignores any override of the PS by the host and returns the configured size of the PS.

Blank parameters

The following session parameters affect Function 5, “Copy Presentation Space”; Function 8, “Copy Presentation Space to String”; and Function 34, “Copy Field to String.”

BLANK (default)
Null characters are converted to spaces (returned as X'20').

NOBLANK
Null characters are not converted (returned as X'00').

Return parameters

Parameters accepted

Function replaces the value of call parameter Data length with the number of session parameters that were set.

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
2	One or more parameter names were not recognized; all recognized parameters were accepted.
9	A system error occurred.

Example

```
/* Set session parameters */
int HllFunc = 9;
char HllDataStr[20];
strcpy (HllDataStr,"SRCHFROM,SRCHFRWD");
/* Length of parameter string */
int HllDataLgth = 17;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 10: Query Sessions

This function returns summary information about each currently started session. The information is returned in a 12-byte data string for each session.

Prerequisites

None.

Applicable session parameters

The following session parameters from Function 9 affect this function.

NOCFGSIZE
The function returns the current size of the connected PS.

CFGSIZE (default)
The function returns the configured size of the PS, ignoring any host overrides

.

Call parameters

An application program must pass the following parameters when calling this function:

Function	10
Data string	A pre-allocated string of 16 bytes per configured session.
Data length	16 bytes per configured session with a maximum of 416 bytes (26 maximum allowable active sessions x 16 bytes).
PS position	Reserved.

Return parameters

Session information

Function replaces content of call parameter Data string with information about currently-open sessions, twelve bytes per session, as follows:

Byte	Description
1	Session short name.
2-4	Reserved
5–12	Session long name.
13	A fixed constant byte value (‘H’).
14	Reserved
15–16	PS size in binary.

Sessions started

Function replaces the value of call parameter Data length with the number of started sessions for which information was returned.

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
2	String length specified was not valid.
9	A system error occurred.

Example

```
int HllFunc = 10;
/* 12 bytes per session, max. 26 sessions */
char HllDataStr[312];
int HllDataLgth = 312;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Tip

Text returned as the session "long name" will be the first eight characters of the name of the configuration ("*.EDP") file used to open the session:

Function 11: Reserve

This function locks the currently connected PS. Locking the PS prevents another application program or terminal operator from entering data into it. Once the PS is locked, it is not accessible until it is unlocked.

The PS can be unlocked with Function 12, “Release”; Function 21, “Reset System”; Function 2, “Disconnect Presentation Space”; or Function 1, “Connect Presentation Space.” Function 1 performs an implicit disconnect. (Terminating a session with Task Manager also unlocks it.)

This function is useful for preventing users from gaining access to the session while an application program sends a series of transactions to the host.

Prerequisites

Function 1, “Connect Presentation Space.”

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	11
Data string	Not applicable.
Data length	Not applicable.
PS position	Reserved.

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
1	The application is not connected to a valid PS.
5	Presentation space cannot be used.
9	A system error occurred.

Example

```
int HllFunc = 11;
char HllDataStr[];
int HllDataLgth;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 12: Release

This function unlocks a PS that was reserved using Function 11, “Reserve.” The target is the currently connected PS.

Release also occurs automatically when the client application program calls Function 2, “Disconnect Presentation Space”; Function 1, “Connect Presentation Space”; Function 21, “Reset System”; or terminates, or the session itself is terminated.

Because release occurs automatically on disconnect, it is not crucial that you use the Release function whenever you end an application.

Prerequisites

Function 1, “Connect Presentation Space.”

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	12
Data string	Not applicable.
Data length	Not applicable.
PS position	Reserved.

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
1	The application is not connected to a valid PS.
9	A system error occurred.

Example

```
int HllFunc = 12;
char HllDataStr[];
int HllDataLgth;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 13: Copy OIA

This function returns the contents of the OIA from the currently connected PS. The length of the OIA data does not change with the terminal model. . Refer to Appendix C for information on interpreting the contents of returned OIA data

Prerequisites

Function 1, "Connect Presentation Space."

Applicable session parameters

The following session parameters from Function 9 affect this function.

OLDOIA (default)
OIA data are returned in EBCDIC

NEWOIA
OIA data are returned in ASCII.

Call parameters

The following session parameters from Function 9 affect this function.

Function	13
Data string	A pre-allocated 104-byte data string
Data length	104
PS position	Reserved.

Return parameters

OIA data

Function replaces content of call parameter Data string with data from the OIA for the currently-connected PS, organized as follows:

Byte	Description
1	The OIA Format Byte for the host access program.
2–81	These bytes contain the untranslatable image of the OIA in hexadecimal codes.
82–103	The OIA bit group.
104	Reserved

Detailed explanation of information contained in this string is given in **Appendix C, "Interpreting the Returning Data String for Function 13."**

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	OIA data were copied; PS is unlocked.
1	The application is not connected to a valid PS.
2	Data string length specified was not valid.
4	OIA data were copied, but the PS is busy.
5	OIA data were copied, but the keyboard is locked.
9	A system error occurred.

Example

```
int HllFunc = 13;
char HllDataStr[103];
/* Length of allocated data area */
int HllDataLgth = 103;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 14: Query Field Attribute

This function returns the field attribute byte for the PS position.

The returning parameter contains the field attribute for the specified PS position. The value of the attribute byte is C0-DF (unprotected field attributes) and E0-FF (protected attributes). A zero attribute means that no field attribute was found in the PS.

Prerequisites

Function 1, "Connect Presentation Space."

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	14
Data string	Not applicable.
Data length	Not applicable.
PS position	The PS position for which field information is wanted

.

Return parameters

Attribute value

Function replaces the value of call parameter Data length with the attribute byte for the specified field. If zero, the PS is unformatted and no attribute can be returned.

3270 Field attribute

Bit	**Meaning**
0-1	Both = 1, field attribute value
2	0 = unprotected; 1 = protected
3	0 = alphanumeric; 1 = numeric only
4-5	00 = normal intensity, not pen detectable
	01 = normal intensity, pen detectable
	10 = high intensity, pen selectable
	11 = nondisplay, not pen detectable
6	Reserved
7	0 = field has not been modified; 1 = field has been modified

Attribute value, continued

5250 Field attribute

Bit	**Meaning**
0	0 = nonfield attribute; 1 = field attribute
1	0 = nondisplay; 1 = display
2	0 = unprotected; 1 = protected
3	0 = normal intensity; 1 = high intensity
4-6	000 = alphameric data; all characters available
	001 = alphabetic only, u/c and l/c, comma, period, hyphen, blank and Dup available
	010 = numeric shift; automatic shift for number
	011 = numeric only: 0-9, comma, period, plus, minus, blank and Dup available
	101 = numeric only: 0-9 or Dup available
	110 = magnetic strip reading device data only
	111 = signed numeric data: 0-9, plus, minus and Dup are available
7	0 = field has not been modified; 1 = field has been modified

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	**Description**
0	Function was successful.
1	The application is not connected to a valid PS.
7	An invalid PS position was specified.
9	A system error occurred.
24	The PS was unformatted.

Example

```
int HllFunc = 14;
char HllDataStr[];
int HllDataLgth;
/* Query field attribute at this position */
int PsPos = 199;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 15: Copy String to Presentation Space

This function copies a string directly into the currently connected PS at the specified location. When the copy operation is complete, the cursor's physical location remains unchanged.

The data string to be copied cannot be any larger than the size of the designated writable area or field. Unprintable characters in the string are translated into blanks in the host system session.

Prerequisites

Function 1, "Connect Presentation Space."

Applicable session parameters

The following session parameters from Function 9 affect this function.

EAB
Extended Attribute Bytes are copied. Two characters are copied from the application data string for each position in the PS. The EAB is the second character.

NOEAB (default)
Extended Attribute Bytes are not present.

XLATE
Extended Attribute Bytes are translated from CGA text mode attributes.

NOXLATE (default)
Extended Attribute Bytes are not translated.

STRLEN (default)
Application must specify the length of the data string to be copied.

STREOT
The string must end to be copied must end with the EOT character.

Call parameters

An application program must pass the following parameters when calling this function:

Function	15
Data string	ASCII text to be copied into the PS. Last byte of the string is EOT if session parameter STREOT is set.

Refer to Appendix D, "Extended Attributes," for information on EAB format.

Data length	Data string length if session parameter STRLEN is set, else not applicable.
PS position	Position of the PS where function is to begin copying data.

Note: This function cannot send keyboard mnemonics.

Return parameters

Result code

Function replaces the value of call parameter <u>PS position</u> with one of the following codes:

Code	Description
0	Function was successful.
1	The application is not connected to a valid PS.
2	Function was called with an invalid parameter.
5	PS is busy or locked, or the data string contained illegal data. The string was not copied.
6	The string was copied, but truncated at the end of the field or screen.
7	An invalid PS position was specified.
9	A system error occurred.

Example

```
int HllFunc = 15;
char HllDataStr[20];
/* Copy "Hello World" to PS */
strcpy(HllDataStr, "Hello World");
/* Length of string to copy */
int HllDataLgth = 11;
/* Position on host screen where string will start*/
int PsPos = 199;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Tip

To copy data to the current PS position, use Function 7, “Query Cursor Location,” to obtain the PS position, then use that value as the <u>PS position</u> calling parameter of this function.

Result code 6 indicates attempt was made to copy data into a protected field. Before writing application code to use this function, the programmer should check that the location where data are to be copied to the PS is (a) an unprotected field and (b) of sufficient extent to accept all the data to be copied.

Function 18: Pause

This function waits a specified amount of time or until a host-initiated update occurs.

If the client application program is already in a Wait, Pause, Get Key, or synchronous file transfer delay, the request for another delay is rejected.

Prerequisites

Function 23, "Start Host Notification," must be called if the application program uses session parameter IPAUSE."

Applicable session parameters

The following session parameters from Function 9 affect this function.

FPAUSE (default)
The function waits the amount of time specified if session parameter FPAUSE is in effect.

IPAUSE
The function waits until a specified host update occurs if session parameter IPAUSE is set and the application has called Function 23, "Start Host Notification. The application must call Function 24, "Query Host Update," before setting the next pause; otherwise, the next pause will be immediately satisfied by the pending event.

Call parameters

An application program must pass the following parameters when calling this function:

Function	18
Data string	An 8-byte string, the first character of which is the PS short name.
Data length	The pause duration in 1/2-second multiples.
PS position	Reserved.

Return parameters

Result code

Function replaces the value of call parameter <u>PS position</u> with one of the following codes:

Code	Description
0	The pause duration expired.
9	A system error occurred.
26	A host session PS or OIA update has occurred.

Example

```
int HllFunc = 18;
/* Wait for update to occur in session B */
char HllDataStr[] = "B^^^^^^^";
/* Wait for 10 sec. or until interrupted */
int HllDataLgth = 20;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Tip

This function consumes some system resources. A practical maximum duration for the pause is 7200 (60 minutes). The application program should not tie up other resources such as keyboard reservations, session connections, and so forth, before entering a pause.

Function 20: Query System

This function returns information about system state that may be useful for determining the cause of a result code 9 being received from some other function call.

Prerequisites

None.

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	20
Data string	A 36-byte string pre-allocated to receive system information.
Data length	Not applicable (36 bytes is implied).
PS position	Reserved.

Return parameters

System information

Function replaces content of call parameter <u>Data string</u> with information about the system state, organized as follows:

Byte	Description
1	Version number.
2–3	Level number.
4–9	Date (month, date, year).
10–12	Reserved.

13	Hardware base, where
	U=unable to determine.
	Z=interrogate positions 28 and 29 for hardware base

14	Program type where E=Extra!, P=Pcomm
15–16	Reserved.
17–18	Host access software version as an ASCII value.
19	Reserved
20-23	Extended error code 1 (system component, printable ASCII)
24-27	Extended error code 2 (fault code, printable ASCII)
28-29	Reserved
30	Reserved
31-32	NLS type (binary)
33	Monitor type (ASCII)
34-36	Reserved.

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful; the data string was returned.
2	An incorrect string length was specified.
9	A system error occurred.

Example

```
int HllFunc = 20;
char HllDataStr[36];
int HllDataLgth;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 21: Reset System

This function resets session parameters changed in Function 9, "Set Session Parameters," to their default state and releases any reserved sessions. This function also releases any connected PSs, and cancels any keystroke interceptions and host update monitors.

An application can call this function at any time to restore session parameters to default values. This function should always be called just before a PCSHLL application program exits.

Prerequisites
None.

Applicable session parameters
None.

Call parameters
An application program must pass the following parameters when calling this function:

Function	21
Data string	Not applicable.
Data length	Not applicable.
PS position	Reserved.

Return parameters

Result code
Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful; the data string was returned.
9	A system error occurred.

Example
```
int HllFunc = 21;
char HllDataStr[];
int HllDataLgth;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Tip
If a session is visible when this function is called, the session will not be released from memory, though any PCSHLL connection of the application with the session will be disconnected.

Function 22: Query Session Status

This function returns specific information about the specified session. It returns the following information in the data string:

- Short and long names
- Terminal type
- Number of rows and columns in the PS

This function provides more information on individual sessions than the allsessions call (Function 10, "Query Sessions").

Prerequisites

None.

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	22
Data string	A 20-byte string pre-allocated to receive session information, the first byte of which contains • a session short name, or • blank or null, indicating the currently-connected PS, or • asterisk ('*'), indicating the session currently with keyboard focus
Data length	20
PS position	Reserved.

Return parameters

Session information

Function replaces content of call parameter Data string with information about the session, organized as follows:

Byte	Description
1	Session short name.
2–4	Reserved
5-12	Session long name.
13	Session type: ‘D’ = 3270 Host ‘F’ = 5250 Host ‘E’ = 3270 Printer ‘G’ = 5250 Printer
14	Session characteristics: Bit 0: 0=No EAB; 1=EABs Bit 1: 0=No programmed symbols 1=Programmed symbols Bit 2–7: Reserved
15–16	Number of rows (binary).
17–18	Number of columns (binary).
19–20	Host code page (binary).

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful; the data string was returned.
1	An invalid session short name was specified.
2	An invalid string length was sent to the function.
9	A system error occurred.

Example

```
int HllFunc = 22;
char HllDataStr[20];
/* Request status of session with keyboard focus */
HllDataStr[0] = '*';
/* Length of data string */
int HllDataLgth = 20;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 23: Start Host Notification

This function begins the process by which PCSHLL determines if the host session PS or OIA has been updated. Your application can then call Function 24, “Query Host Update,” to find out more specific information about the update. This function also enables the designated host session event to end an interruptible pause started with Function 18, “Pause.”

Prerequisites

None.

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	23
Data string	A 16-byte string. (See format below).
Data length	16
PS position	Reserved.

Data string format

Byte	Description
1	Session short name. If blank or null, the session to which the application is currently connected.
2-4	Reserved
5	One of the following characters: “P”—Notification of PS update “O”—Notification of OIA update “B”—Notification of both OIA and PS updates “A” —Asynchronous-mode notification requested. “M” —Asynchronous message-mode notification requested.
6-8	Reserved
9-12	If byte 5 contains “A”, the 32-bit address of an event object. If byte 5 contains “M”, RegisterWindowMessage("PCSHLL”) return value, not zero, in bytes 9 and 10. (Bytes 11 and 12 are ignored.) If byte 5 contains neither “A” nor “M”, reserved.
13	If byte 5 contains “A” or “M”, one of the following characters: “P”—Notification of PS update “O”—Notification of OIA update “B”—Notification of both OIA and PS updates If byte 5 contains neither “A” nor “M”, reserved.
14-16	Reserved

Return parameters

Asynchronous notification outcome

When "A" appears in byte 5 of call parameter Data string, function replaces the value of call parameter Data string with the following information:

Byte	Description
1	Session short name.
2–8	Reserved
9-12	Address of the event object returned by PCSHLL. The application can wait on this event

Asynchronous message mode outcome

When "M" appears in byte 5 of call parameter Data string, function replaces the value of call parameter Data string with the following information:

Byte	Description
1	Session short name.
2–8	Reserved
9-10	Task ID of asynchronous message mode.

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
1	An invalid session short name was specified.
2	An invalid parameter was specified.
9	A system error occurred.

Example

```
int HllFunc = 23;
char HllDataStr[16];
/* Short name of session */
HllDataStr[0] = 'E';
/* Both OIA and PS updates */
HllDataStr[4] = 'B';
/* Host event buffer length */
int HllDataLgth = 16;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 24: Query Host Update

This function allows your application to determine if the host has updated the PS or OIA since the last time Function 23, “Start Host Notification” or this function was called.

The application program need not be connected to the PS for updates; however, it must specify the short name for the desired session.

Prerequisites

Function 23, “Start Host Notification.”

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	24
Data string	A four-byte string, the first character of which is the short name of the desired session, or blank or null requesting the connected session.
Data length	Not applicable (4 is implied).
PS position	Reserved.

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	**Description**
0	No updates occurred since the last call.
1	An invalid PS was specified.
8	Function 23, “Start Host Notification,” has not been called for this PS.
9	A system error occurred.
21	The OIA was updated.
22	The PS was updated.
23	Both OIA and PS were updated.

Example

```
int HllFunc = 24;
char HllDataStr[4];
/* Short name of session */
HllDataStr[0] = 'B';
int HllDataLgth;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 25: Stop Host Notification

This function disables the capability of Function 24, “Query Host Update.” This function can also be used to stop host events from affecting Function 18, “Pause.”

Prerequisites

Function 23, “Start Host Notification.”

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	25
Data string	A four-byte string, the first character of which is the short name of the desired session, or blank or null requesting the connected session.
Data length	4
PS position	Reserved.

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	**Description**
0	The stop notification was successful.
1	An invalid session short name was specified.
8	Function 23, “Start Host Notification,” has not been called for this PS.
9	A system error occurred.

Example

```
int HllFunc = 25;
char HllDataStr[4];
/* Short name of session */
HllDataStr[0] = 'B';
int HllDataLgth = 4;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 30: Search Field

This function searches through a specified field of the currently connected PS for a specified string. It can be used to search for a string in either protected or unprotected fields of a field formatted host PS. If the target string is found, this function returns the starting position of the string.

This search is always case-sensitive. This function requires a complete match of target string to field contents, regardless of the direction of the search.

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

Prerequisites

Function 1, "Connect Presentation Space."

Applicable session parameters

The following session parameters from Function 9 affect this function.

SRCHALL (default)
The entire field containing the specified PS position is searched.

SRCHFROM
Search begins at the specified position in the field.

SRCHFRWD (default)
Search finds first instance of the string between the origin and the end of the field.

SRCHBKWD
Search finds the last instance of the string between the field origin and the end of the field, or the specified PS position (if SRCHFROM is set).

STRLEN (default)
Application must specify the length of the data string to be copied.

STREOT
The string must end to be copied must end with the EOT character.

Call parameters

An application program must pass the following parameters when calling this function:

Function	30
Data string	ASCII text to be searched for in the field. Last byte of the string is EOT if session parameter STREOT is set.
Data length	Data string length if session parameter STRLEN is set, else not applicable.
PS position	Specifies a PS position within the target field or on the field attribute that begins it.. For SRCHALL, this can be any PS position within the field. For SRCHFROM, search begins here for SRCHFRWD or ends here for SRCHBKWD.

Return parameters

PS Position

Function replaces the value of call parameter Data length with the PS position where the specified text was found. If zero, the specified text was not found.

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful; the string was found.
1	The application is not connected to a valid PS.
2	The string length was zero; or, if STREOT was in effect, no EOT character was found in the given search string.
7	An invalid PS position was specified.
9	A system error occurred.
24	The string was not found.

Example

```
int HllFunc = 30;
char HllDataStr[100];
/* Target string to search for "Hello" */
strcpy (HllDataStr, "Hello");
int HllDataLgth = 5;
int PsPos;
/* Start position for search */
PsPos = 199;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 31: Find Field Position

This function searches through the currently connected PS for a field's beginning position and returns the position. This function will search for either protected or unprotected fields, but the fields must be in a field-formatted host PS.

Prerequisites

Function 1, "Connect Presentation Space."

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	31
Data string	A 2-character code specifying the field to find. (See format below.)
Data length	Not applicable (2 is implied).
PS position	A position in the PS that lies within the field or on the field attribute that begins it.

Data string format

Content	Description
^^ or T^	This field.
N^	Next field (protected or unprotected).
NP	Next protected field.
NU	Next unprotected field.
P^	Previous field (protected or unprotected)
PP	Previous protected field.
PU	Previous unprotected field.

^ = a space

Return parameters

Field position

Function replaces the value of call parameter Data length with the PS position where the specified field begins. If zero, the field is either zero length or the PS is unformatted.

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful; the field was found.
1	The application is not connected to a valid PS.
2	An incorrect parameter was specified.
7	An invalid PS position was specified.
9	A system error occurred.
24	Either the field was not found, or the PS was unformatted.
28	The field length is zero bytes.

Example

```
int HllFunc = 31;
char HllDataStr[10];
/* Find start position of this field (t, space) */
strcpy (HllDataStr, "T ");
int HllDataLgth;
/* Start search at PS position */
int PsPos = 199;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 32: Find Field Length

This function returns the length of a specified PS field, protected or unprotected, and is the number of characters contained in the field between the attribute byte that begins the field and the next-following field attribute.

NOTE. This function wraps from the end to the beginning of the PS.

Prerequisites

Function 1, "Connect Presentation Space."

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	32
Data string	A 2-character code specifying the field to find. (See format below.)
Data length	Not applicable (2 is implied).
PS position	A position in the PS that lies within the field or on the field attribute that begins it.

Data string format

Content	Description
^^ or T^	This field.
N^	Next field (protected or unprotected).
NP	Next protected field.
NU	Next unprotected field.
P^	Previous field (protected or unprotected)
PP	Previous protected field.
PU	Previous unprotected field.

^ = a space

Return parameters

Field length

Function replaces the value of call parameter <u>Data length</u> with the length of the specified field. If zero, the field was not found, or is zero length, or the PS is unformatted.

Note If a field attribute is followed by another field attribute, the field is assumed to have a length of zero.

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful; the field was found.
1	The application is not connected to a valid PS.
2	An incorrect parameter was specified.
7	An invalid PS position was specified.
9	A system error occurred.
24	Either the field was not found, or the PS was unformatted.

Example

```
int HllFunc = 32;
char HllDataStr[10];
/* Find length of this field (t, space) */
strcpy (HllDataStr, "T ");
int HllDataLgth;
/* Start search at PS position */
int PsPos = 199;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 33: Copy String to Field

This function copies characters to a specific unprotected field in a field-formatted PS.

The copy operation ends when one of four conditions is met:

- The entire string has been copied.
- The text has been written to the last field position.
- The function has copied the specified number of characters in the data length parameter.
- The character before the EOT character is copied when EOT is specified.

Note AID key character sequences are not evaluated when using this function. They will be copied to the field as literal strings. Function 3, "Send Key," must be used to send an AID key to a session.

Prerequisites

Function 1, "Connect Presentation Space."

Applicable session parameters

The following session parameters from Function 9 affect this function.

EAB
Text and EABs are copied from the data string.

NOEAB (default)
The data string does not contain EABs.

XLATE
EABs are translated from CGA text mode attributes.

NOXLATE (default)
EABs are copied without translation.

STRLEN (default)
Application must specify the length of the data string to be copied.

STREOT
The string must end to be copied must end with the EOT character.

Call parameters

An application program must pass the following parameters when calling this function:

Function	33
Data string	ASCII text to be copied into the field. Last byte of the string is EOT if session parameter STREOT is set.

Refer to Appendix D, "Extended Attributes," for information on EAB formats.

Data length	Data string length if session parameter STRLEN is set, else not applicable.
PS position	A position in the PS that lies within the field or on the field attribute that begins it. Copy always starts at the beginning of the field.

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	**Description**
0	Success; the string was copied to the target field in the PS.
1	The application is not connected to a valid PS.
2	A string length of zero was specified.
5	Either the target field was protected or inhibited; or a nondisplayable character was included in the string.
6	The string was copied, but it was truncated because the field was shorter than the string.
7	An invalid PS position was specified.
9	A system error occurred.
24	The host PS is unformatted.

Example

```
int HllFunc = 33;
char HllDataStr[20];
/* Copy "Hello World" to field */
strcpy (HllDataStr, "Hello World");
/* Length of string to copy */
int HllDataLgth = 11;
/* Copy to field containing this position */
int PsPos = 199;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 34: Copy Field to String

This function copies all characters from a field in the currently connected PS into a string. It can be used with either protected or unprotected fields, but only in a field-formatted PS.

The copy operation begins at the field's origin. This position and length information can be found by using Function 31, "Find Field Position," and Function 32, "Find Field Length."

This function ends when one of two conditions is met:

- The last character in the field was copied.
- All character positions in the copy string have been filled.

Prerequisites

Function 1, "Connect Presentation Space."

Applicable session parameters

The following session parameters from Function 9 affect this function.

EAB
Text and EABs are copied to the buffer.

NOEAB (default)
EABs are not copied.

XLATE
EABs are translated to CGA text mode attributes.

NOXLATE (default)
EABs are not translated.

DISPLAY (default)
Non-display text is copied to the target buffer in the same manner as the display data.

NODISPLAY
Non-display text is copied to the target buffer as nulls.

Call parameters

An application program must pass the following parameters when calling this function:

Function	34
Data string	The string to which the program copies the contents of the field.
Data length	The number of characters to be copied.

NOTE: Data string must be at least twice this length if the EAB session parameter is set.

PS position	A position in the PS that lies within the field or on the field attribute that begins it. Copy starts at the beginning of the field.

Return parameters

Field content

Function replaces content of call parameter Data string with text from the field.

Refer to Appendix D, “Extended Attributes,” for information on EAB interpretation.

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	Success; text from the field has been copied to data string.
1	The application was not connected with a host PS.
2	A parameter error was detected.
6	The string was copied, but it was truncated because the string was shorter than the field.
7	An invalid PS position was specified.
9	A system error occurred.
24	The presentation space is unformatted.

Example

```
int HllFunc = 34;
/* Allocated data buffer */
char HllDataStr[10];
/* Length of allocated buffer */
int HllDataLgth = 10;
/* Copy from field containing this position */
int PsPos = 199;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 40: Set Cursor

This function sets the cursor position to the target PS position in the currently connected PS.

Prerequisites

Function 1, “Connect Presentation Space.”

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	40
Data string	Not applicable.
Data length	Not applicable.
PS position	The desired cursor position in the PS.

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	Success; text from the field has been copied to data string.
1	The application was not connected with a host PS.
4	The PS is busy.
7	An invalid PS position was specified.
9	A system error occurred.

Example

```
int HllFunc = 40;
char HllDataStr[];
int HllDataLgth;
/* Where to put cursor */
int PsPos = 199;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 41: Start Close Intercept

This function allows the application to intercept user-generated close requests. The function intercepts and discards the close request until the client application program requests Function 43, “Stop Close Intercept.” Multiple client application programs can request this function for the same session.

Prerequisites

None.

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	41
Data string	A 12-byte string. (See format below).
Data length	12
PS position	Reserved.

Data string format

Byte	Description
1	Session short name. If blank or null, the session to which the application is currently connected.
2-4	Reserved
5	“M” —Asynchronous message-mode notification requested. Any other value, asynchronous message mode not requested.
6-8	Reserved
9-12	If byte 5 contains “M”, RegisterWindowMessage("PCSHLL”) return value, not zero. (Bytes 11 and 12 are ignored.) If byte 5 does not contain “M”, bytes 9-12 are reserved.

Return parameters

Asynchronous notification outcome

When a value other than "M" appears in byte 5 of call parameter Data string, function replaces the value of call parameter Data string with the following information:

Byte	Description
1	Session short name.
2–8	Reserved
9-12	Address of the event object returned by PCSHLL. The application can wait on this event

Asynchronous message mode outcome

When "M" appears in byte 5 of call parameter Data string, function replaces the value of call parameter Data string with the following information:

Byte	Description
1	Session short name.
2–8	Reserved
9-10	Task ID of asynchronous message mode.

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
1	An invalid PS was specified.
2	A string length of zero was specified.
9	A system error occurred.
10	Function is not supported by the emulator.

Example

```
int HllFunc = 41;
char HllDataStr[12];
/* Short name of session */
HllDataStr[0] = 'B';
HllDataStr[5] = 0;
int HllDataLgth = 12;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 42: Query Close Intercept

This function allows the application to determine if a user-generated close request has been issued.

Prerequisites

Function 41, “Start Close Intercept.”

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	42
Data string	A four-byte text string of which the first character is the session short name.
Data length	4
PS position	Reserved.

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	No close intercept event has occurred.
1	An invalid PS was specified.
2	A string length of zero was specified.
8	No call to Function 41, “Start Close Intercept,” was issued
9	A system error occurred.
12	The session was stopped.
26	A close intercept event has occurred since the last call to this function.

Example

```
int HllFunc = 42;
char HllDataStr[4];
/* Short name of session */
HllDataStr[0] = 'B';
int HllDataLgth = 4;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 43: Stop Close Intercept

This function ends the intercept started by Function 41, “Start Close Intercept.” Once the application calls this function, all user generated close requests are processed in the normal way.

Prerequisites

Function 41, “Start Close Intercept.”

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	43
Data string	A four-byte text string of which the first character is the session short name.
Data length	4
PS position	Reserved.

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
1	An invalid PS was specified.
2	A string length of zero was specified.
8	Function 41, “Start Close Intercept,” was not called prior to this function.
9	A system error occurred.
12	The session was stopped.

Example

```
int HllFunc = 43;
char HllDataStr[1];
/* Short name of session */
HllDataStr[0] = 'B';
int HllDataLgth = 4;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 50: Start Keystroke Intercept

This function allows an 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 buffer overflows or call is made to Function 21, "Reset System," or Function 53, "Stop Keystroke Intercept."

Intercepted keystrokes can be

- received through Function 51, "Get Key," and sent to the same or another session with Function 3, "Send Key"
- accepted or rejected through Function 52, "Post Intercept Status"
- replaced by other keystrokes with Function 3, "Send Key"
- used to trigger other processes.

If AID-key-only intercept is requested (option "D" is specified), non-AID keys will be sent to the PS and only AID keys will be available to the application.

Note Extended processing of each keystroke may cause unacceptable delays for keyboard users.

Prerequisites

None.

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	50
Data string	A 16-byte structure specifying the intercept desired. (See format below.)
Data length	16
PS position	Reserved.

Data string format

Byte	Description
1	Session short name. If blank or null, the session to which the application is currently connected.
2-4	Reserved
5	One of the following characters: D for AID keystrokes only L for all keystrokes M for asynchronous message mode notification
9-12	If byte 5 contains "M", RegisterWindowMessage("PCSHLL") return value, not zero. (Bytes 11 and 12 are ignored.) If byte 5 does not contain "M", bytes 9-12 are reserved.
13	If byte 5 contains "M", one of the following characters: D for AID keystrokes only L for all keystrokes If byte 5 does not contain "M", ignored.
14-16	Reserved.

Return parameters

Result code

Function replaces the value of call parameter <u>PS position</u> with one of the following codes:

Code	Description
0	The function was successful.
1	An invalid PS was specified.
2	An invalid option was specified.
4	The PS is busy.
9	A system error occurred.

Example

```
int HllFunc = 50;
char HllDataStr[16];
/* Short name of session */
HllDataStr[0] = 'B';
/* All keystrokes */
HllDataStr[4] = 'D';
/* Event buffer space */
int HllDataLgth = 16;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 51: Get Key

This function allows your application to receive the keystrokes for the sessions that were specified with Function 50, "Start Keystroke Intercept."

Use Function 3, "Send Key," to pass keystrokes to the target PS.

When keystrokes are available, they are read into the data area that you have provided in your client application program. Each keystroke is represented by one of the key codes listed in **Appendix B, "Keyboard Mnemonics."**

The CAPSLOCK key on the PC works like the SHIFTLOCK key on the host system; it produces the uppercase of all keys, not just alphanumeric keys. So if the application is getting keys with CAPSLOCK on, it gets all keys in the shifted state.

Prerequisites

Function 50, "Start Keystroke Intercept."

Applicable session parameters

The following session parameters from Function 9 affect this function.

TWAIT (default)
The function does not return control to the calling application until a key has been intercepted.

LWAIT
The function does not return control to the calling application until a key has been intercepted.

NWAIT
The function checks for intercepted keystrokes and returns immediately.

Call parameters

An application program must pass the following parameters when calling this function:

Function	51
Data string	A 12-character code specifying the intercept desired. (See format below.)
Data length	12
PS position	Reserved.

Data string format

Byte	Description
1	Session short name. If blank or null, the session to which the application is currently connected.
2-4	Reserved
5-11	Blanks reserving space for the intercepted data.
12	Reserved.

Return parameters

Intercept string

Function replaces content of call parameter Data string with information describing the keystroke intercepted.

Byte	Description
1	A 1-character session short name; if or blank/null indicating a intercept is for the currently connected PS.
2-4	Reserved
5	Code character, one of the following: • A= ASCII returned • M= Keystroke mnemonic
6-11	Allocated buffer used for queuing and dequeuing keystrokes. This buffer contains the following: • If the key returned is a character key, bytes 6 and 7 contain the ASCII character followed by 00. If it is a 3270 key, bytes 6 and 7 will contain a mnemonic for the keystroke (for example, @5 represents PF5). • Bytes 8 through 11 contain nulls, unless the key returned was a combination key such as ERASEINPUT, for which bytes 6 through 9 would contain @A@F and bytes 10-11 nulls.

Typical intercept strings

Intercept strings Function 51 might return are shown below with their keyboard equivalents

Intercept string	Keyboard equivalent
E At	"E" represents the session and "A" informs your PCSHLL application that the keystrokes will be received as ASCII; the returning key is a lowercase "t" (Bytes 6-7 = X'00').
E M@2	"E" represents the session, "M" indicates that the keystrokes will be returned as key mnemonics, and "@2" indicates the key being returned is PF2 (Bytes 8–9 = X'00').
E M@A@Q	"E" represents the session, "M" indicates the keystrokes will be returned as mnemonics, and "@A@Q" indicates that the key being received is ATTN. (Bytes 10–11 = X'00').

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
1	An invalid PS was specified.
5	Inhibited to non-AID keys.
8	Function 50, "Start Keystroke Intercept," was not called prior to calling this function for the specified PS.
9	A system error occurred.
20	An undefined keystroke combination was entered.
25	The requested keystrokes are not available.
31	The keystroke queue overflowed and keystrokes were lost.

Example

```
int HllFunc = 51;
/* Allocate space for returned key */
char HllDataStr[12];
/* Short name of session */
HllDataStr[0] = 'B';
int HllDataLgth = 12;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 52: Post Intercept Status

This function places a sentinel on keyboard input that sounds a beep if the keystroke obtained through Function 51, "Get Key," was rejected.

Prerequisites

Function 50, "Start Keystroke Intercept."

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	52
Data string	An 8 byte string specifying status to be posted. (See format below.)
Data length	8
PS position	Reserved.

Data string format

Byte	Description
1	Session short name. If blank or null, the session to which the application is currently connected.
2-4	Reserved
5	One of the following: • A for accepted keystrokes • R for rejected keystrokes
6-8	Reserved

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
1	An invalid PS was specified.
2	An invalid option was specified.
8	Function 50, "Start Keystroke Intercept," was not called prior to calling this function for the specified PS.
9	A system error occurred.

Example

```
int HllFunc = 52;
char HllDataStr[8];
/* Short name of session */
HllDataStr[0] = 'B';
/* Rejected keystroke */
HllDataStr[4] = 'R';
int HllDataLgth;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 53: Stop Keystroke Intercept

This function ends an application's ability to intercept keystrokes for the specified session.

Prerequisites

Function 50, "Start Keystroke Intercept."

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	53
Data string	A 4-byte string of which the first character is the session short name or a blank/null character indicating a request for the currently connected PS.
Data length	4
PS position	Reserved.

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
1	An invalid session short name was specified.
8	Function 50, "Start Keystroke Intercept," was not called prior to calling this function for the specified PS.
9	A system error occurred.

Example

```
int HllFunc = 53;
char HllDataStr[4];
/* Short name of session */
HllDataStr[0] = 'B';
int HllDataLgth = 4;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 60: Lock Presentation Space API

This function allows one application to obtain or release exclusive control of the presentation space window. When locked, no other application can execute functions requiring interaction with the presentation space -- specifically, PCSHLL functions 1 (if session parameter CONPHYS is in effect), 3, 5, 6, 8, 11, 12, 14, 15, 30, 31, 32, 33, 34, 40, 90 and 91.

While one application has the presentation space locked, certain PCSHLL functions requested from other applications are queued until the controlling application unlocks the presentation space. At that time, any queued requests are processed in FIFO order.

Prerequisites

Function 1, "Connect Presentation Space."

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	60
Data string	An 8 byte string specifying status to be posted. (See format below.)
Data length	8
PS position	Reserved.

Data string format

Byte	Description
1	Session short name.
2-4	Reserved
5	One of the following: • L to lock the API • U to unlock the API
6	One of the following: • R to return if the API is already locked by an application • Q to queue the LOCK request if the API is already locked
7-8	Reserved

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
1	An invalid session short name was specified.
2	Parameters error was encountered
9	A system error occurred.
43	The API is already locked by another application, or UNLOCK failed because the API was not previously locked.

Example

```
int HllFunc = 60;
char HllDataStr[8];
/* Short name of session */
HllDataStr[0] = 'A';
/* Request LOCK */
HllDataStr[4] = 'L';
/* and return if already locked */
HllDataStr[5] = 'R';
int HllDataLgth = 8;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 61: Lock Window Services API

This function allows one application to obtain or release exclusive control of the presentation space window. When locked, no other application can modify the presentation space window state using functions 104 or 106.

While one application has the presentation space locked, certain PCSHLL functions requested from other applications are queued until the controlling application unlocks the presentation space. At that time, any queued requests are processed in FIFO order.

Prerequisites
Function 101, "Connect Window Services."

Applicable session parameters
None.

Call parameters
An application program must pass the following parameters when calling this function:

Function	60
Data string	An 8 byte string specifying status to be posted. (See format below.)
Data length	8
PS position	Reserved.

Data string format

Byte	Description
1	Session short name.
2-4	Reserved
5	One of the following: • L to lock the API • U to unlock the API
6	One of the following: • R to return if the API is already locked by an application • Q to queue the LOCK request if the API is already locked
7-8	Reserved

Return parameters

Result code
Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
1	An invalid session short name was specified.
2	Parameters error was encountered
9	A system error occurred.
43	The API is already locked by another application, or UNLOCK failed because the API was not previously locked.

Example

```
int HllFunc = 61;
char HllDataStr[8];
/* Short name of session */
HllDataStr[0] = 'A';
/* Request LOCK */
HllDataStr[4] = 'L';
/* and return if already locked */
HllDataStr[5] = 'R';
int HllDataLgth = 8;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 90: Send File

This function allows the client application program to send a file to a host session.

PCSHLL-initiated file transfers are synchronous, returning control on completion of the file transfer.

The program requesting synchronous file transfers must not be intercepting keystrokes for any sessions, must not be awaiting the outcome of another synchronous file transfer, and must not be waiting for host events in any session.

Prerequisites

The session to be used for a file transfer must be logged on and at a host system prompt.

Applicable session parameters

The following session parameters from Function 9 affect this function.

NOQUIET (default)
SEND messages are displayed

QUIET
SEND messages are not displayed.

STRLEN (default)
Strings are passed with an explicit length. The client application program provides the value.

STREOT
All strings are passed with the character specified in the EOT session parameter denoting the string end instead of the explicit length.

EOT= char
This character denotes the end of a string when the STREOT session parameter has been set. Null (/0) is the default value.

TIMEOUT= char
The character specifies how many 30-second cycles may elapse before CTRL BREAK is issued.

Call parameters

An application program must pass the following parameters when calling this function:

Function	90
Data string	A string (maximum 255 bytes) containing the send command string. Last byte of the string is EOT if session parameter STREOT is set.
Data length	Data string length if session parameter STRLEN is set, else not applicable.
PS position	Reserved.

Return parameters

Result code

Function replaces the value of call parameter <u>PS position</u> with one of the following codes:

Code	Description
2	A parameter error occurred.
3	The file was transferred.
4	The file was transferred with records segmented.
5	Workstation file name not valid or file not found.
9	A system error occurred.
27	The file transfer was terminated by CTRL C.
301	Invalid function number.
302	File not found.
303	Path not found.
305	Access denied.
308	Insufficient memory.
310	Invalid environment.
311	Invalid format.

Example

```
int HllFunc = 90;
/* Send command string   Assumes        */
/*    PC filename = pcfile.ext          */
/*    Session short name = D            */
/*    Host filename = hostfile.ext      */
/*    CMS transfer options = ASCII,CRLF */
char HllDataStr [] = "pcfile.ext d: hostfile ext (ASCII CRLF";
/* Length of data string */
int HllDataLgth = strlen(HllDataStr);
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 91: Receive File

This function allows the client application program to receive a file from a host session.

PCSHLL-initiated file transfers are synchronous, returning control on completion of the file transfer.

The program requesting synchronous file transfers must not be intercepting keystrokes for any sessions, must not be awaiting the outcome of another synchronous file transfer, and must not be waiting for host events in any session.

Prerequisites

The session to be used for a file transfer must be logged on and at a host system prompt.

Applicable session parameters

The following session parameters from Function 9 affect this function.

NOQUIET (default)
SEND messages are displayed

QUIET
SEND messages are not displayed.

STRLEN (default)
Strings are passed with an explicit length. The client application program provides the value.

STREOT
All strings are passed with the character specified in the EOT session parameter denoting the string end instead of the explicit length.

EOT= char
This character denotes the end of a string when the STREOT session parameter has been set. Null (/0) is the default value.

TIMEOUT= char
The character specifies how many 30-second cycles may elapse before CTRL BREAK is issued.

Call parameters

An application program must pass the following parameters when calling this function:

Function	91
Data string	A string (maximum 255 bytes) containing the receive command string. Last byte of the string is EOT if session parameter STREOT is set.
Data length	Data string length if session parameter STRLEN is set, else not applicable.
PS position	Reserved.

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
2	A parameter error occurred.
3	The file was transferred.
4	The file was transferred with records segmented.
9	A system error occurred.
27	The file transfer was terminated by CTRL C.
301	Invalid function number.
302	File not found.
303	Path not found.
305	Access denied.
308	Insufficient memory.
310	Invalid environment.
311	Invalid format.

Example

```
int HllFunc = 91;
/* Receive command string   Assumes     */
/*    PC filename = pcfile.ext          */
/*    Session short name = B            */
/*    Host filename = hostfile.ext      */
/*    CMS transfer options = ASCII,CRLF */
char HllDataStr [] = "pcfile.ext b: hostfile ext (ASCII CRLF";
/* Length of data string */
int HllDataLgth = strlen(HllDataStr);
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 99: Convert Position or RowCol

This function converts a PS position value into display row/column coordinates or a row/column value into PS position display coordinates.

When the conversion is made, the function considers the model number of the host system display type being emulated. This function does not change the cursor position.

Prerequisites

None.

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	99
Data string	An 8-byte string containing the conversion request. (See format below.)
Data length	If “R” is specified in the data string, the data length specifies the row number. If “P” is specified in the data string, the data length is not applicable.
PS position	If “P” is specified in the data string, this must specify a valid PS position. If “R” is specified in the data string, this must specify a valid column number (1 to 132).

Data string format

Byte	Description
1	A 1-character session short name.
2-4	Reserved
5	One of the following: • “P” to convert from PS position to row-column coordinates. • “R” to convert from row-column coordinates to PS position.
6-8	Reserved

Return parameters

Row number

If converting PS position to row-column coordinates, function replaces the value of call parameter Data length with the row number for the PS position. If the value returned is zero, the row number is invalid for the PS.

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	An invalid PS position or column was specified.
>0	The PS position or column number, depending on the type of conversion being performed.
9998	An invalid session short name was specified.
9999	Second character in data string was not an uppercase “P” or “R.”

Example

```
int HllFunc = 99;
/* Short name of session */
char HllDataStr[8];
HllDataStr[0] = 'B';
/* Convert position to row column */
HllDataStr[4] = 'P';
int HllDataLgth;
/* Convert this position to row column */
PsPos = 199;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 101: Connect Window Services

This function allows a PCSHLL application to connect to and manage the PS window.

A PCSHLL application can be connected to more than one PS window at the same time. The application can switch between windows without having to disconnect.

Only one PCSHLL application can be connected to a PS window at any one time. Another application can access the PS window only if the first application exits the connection or switches to another PS window connection.

Function 21, “Reset System,” reinitializes the PCSHLL application to its starting point.

Prerequisites

None.

Applicable session parameters

The following session parameters from Function 9 affect this function.

WRITE_SUPER (default)
This parameter is set by a client application program that requires write access and allows only supervisory applications to connect to its PS.

WRITE_WRITE
This parameter is set by a client application program that requires write access and allows other applications that have predictable behavior to connect to its PS.

WRITE_READ
This parameter is set by a client application program that requires write access and allows other applications to use read-only functions on its PS.

WRITE_NONE
This parameter is set by a client application program that requires exclusive access to its PS. No other applications will have access to its PS.

SUPER_WRITE
This parameter is set by a supervisory client application program that allows applications with write access to share the connected PS. The client application program setting this parameter will not cause errors for other applications, but will provide only supervisory-type functions.

WRITE_READ
This parameter is set by a client application program that requires read-only access and allows other applications that perform read-only functions to connect to its PS.

Call parameters

An application program must pass the following parameters when calling this function:

Function	101
Data string	A 4-byte text string of which the first character is the session short name.
Data length	4
PS position	Reserved.

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
1	An invalid session short name was specified.
9	A system error occurred.
10	The function is not supported.
11	The PS was busy.

Example

```
int HllFunc = 101;
char HllDataStr[4];
/* Short name of session */
HllDataStr[0] = 'B';
int HllDataLgth = 4;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 102: Disconnect Window Services

This function disconnects the window services connection between an PCSHLL application and the PS.

Prerequisites

Function 101, "Connect Window Services."

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	102
Data string	A 4-byte text string of which the first character is the session short name.
Data length	4.
PS position	Reserved.

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
1	An invalid session short name was specified.
9	A system error occurred.

Example

```
int HllFunc = 102;
char HllDataStr[4];
/* Short name of session */
HllDataStr[0] = 'B';
int HllDataLgth = 4;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 103: Query Window Coordinates

This function requests the window coordinates of a PS. Window coordinates are returned in pixels.

Prerequisites

Function 101, “Connect Window Services.”

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	103
Data string	A 20-byte data string. (See format below.)
Data length	Not applicable (17 is implied).
PS position	Reserved.

Data string format

Byte	Description
1	Session short name, or blank/null indicating a request for the currently connected PS.
2-20	Reserved

Return parameters

Window coordinates

Function replaces content of call parameter Data string with information about the session window coordinates.

Byte	Description
0	Session short name, or blank/null indicating a request for the currently connected PS.
2-4	Reserved
5–20	Four 32-bit unsigned integers (Xleft, Ybottom, Xright, Ytop) that return the coordinates (in pixels) of a rectangular window relative to the desktop window.

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
1	An invalid session short name was specified.
9	A system error occurred.
12	The host session was stopped.

Example

```
int HllFunc = 103;
char HllDataStr[20];
/* Short name of session */
HllDataStr[0] = 'B';
int HllDataLgth = 20;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 104: Window Status

This function allows the application to query or change the PS window. The application can change the size, location, or visible state of a PS window. The function can return information regarding the size, location, relative placement, and visible state of a PS window.

Prerequisites

Function 101, "Connect Window Services."

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	104
Data string	A 24 or 28-byte data string. (See formats below.)
Data length	24 if extended status is requested; 28 otherwise.
PS position	Reserved.

Data string format, Set status request

Byte	Description
1	A 1-character session short name.
2-4	Reserved
5	X01 – Set status
6	Reserved
7–8	The status set bits. The following codes are valid: • X'0001' — Change window size • X'0002' — Move window • X'0004' — ZORDER window replacement • X'0008' — Set window to visible • X'0010' — Set window to invisible • X'0080' — Activate window • X'0100' — Deactivate window • X'0400' — Minimize window • X'0800' — Maximize window • X'1000' — Restore window
9-12	The X-window position coordinate in pixels. (These bytes are ignored if the move option is not set).
13-16	The Y-window position coordinate in pixels. (These bytes are ignored if the move option is not set).
17-20	The X-window size in pixels. (These bytes are ignored if the size option is not set).
21-24	The Y-window size in pixels. (These bytes are ignored if the size option is not set).
25-28	The window handle for relative window placement. (These bytes are ignored if the ZORDER option is not set.) • X'00000003' — Place window in front of siblings • X'00000004' — Place window behind siblings

Data string format, Query status request

Byte	Description
1	A 1-character session short name.
2-4	Reserved
5	X02 – Query status.
6	Reserved
7-8	X'0000'
9-28	Reserved.

Data string format, Query extended status request

Byte	Description
1	A 1-character session short name.
2-4	Reserved
5	X03 – Query extended status.
6	Reserved
7-8	X'0000'
9-24	Reserved.

Return parameters

Query status result

If the request option (byte 2 of call parameter Data string) was 2 (query status), content of bytes 3 – 16 of call parameter Data string is updated as follows:

Byte	Description
7-8	A word containing a logical OR of bits indicating window state: • 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
9-12	The X-window position coordinate.
13-16	The Y-window position coordinate.
17-20	The X-window size in device units.
21-24	The Y-window size in device units.
25-28	The window handle of the session.

Query extended status result

If the request option (byte 2 of call parameter Data string) was 3 (query extended status), content of bytes 3 – 20 of call parameter Data string is updated as follows:

Byte	Description
7-8	A word containing a logical OR of bits indicating window state: • 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
9-10	The X-dimension font size in pixels.
11-12	The Y-dimension font size in pixels.
13-16	Reserved (always zero)
17-18	The row number of the first visible character of the PS.
19-20	The column number of the first visible character of the PS.
21-24	The window handle of the session.

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
1	An invalid session short name was specified.
2	A parameter error was detected.
9	A system error occurred.
12	The host session was stopped.

Example

```
int HllFunc = 104;
char HllDataStr[20];
/* Query extended status for session B */
HllDataStr[0] = 'B';
HllDataStr[4] = 3;
int HllDataLgth;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Function 106: Change PS Window Name

This function allows the application to change or reset a PS window name.

The exit list processing will reset the name if the application does not do so before exiting. To retain the changed PS name, use Function 102, “Disconnect Window Services.”

Prerequisites

Function 101, “Connect Window Services.”

Applicable session parameters

None.

Call parameters

An application program must pass the following parameters when calling this function:

Function	106
Data string	A 68-byte string. (See format below.)
Data length	68
PS position	Reserved.

The length of call parameter Data string must be specified in the call *and* the data string must end with a null character. If a null character is found before the specified length, the string is truncated at that point and the remaining data are lost. If the specified length is reached and the data does not end with a null character, the last byte of the specified length is replaced with a null character and the rest of the data string is lost.

Data string format

Byte	**Description**
1	1-character session short name
2-4	Reserved
5	One of the following: • X’01’ — Change the PS window name • X’02’ — Reset the PS window name
6–66	An ASCII string of 1 to 61 bytes including terminating null character. For 5250 emulation, at least one non-null character must precede the terminating null.
67-68	Reserved

Return parameters

Result code

Function replaces the value of call parameter PS position with one of the following codes:

Code	Description
0	The function was successful.
1	An invalid session short name was specified.
2	A parameter error was detected.
9	A system error occurred.
12	The host session was stopped.

Example

```
int HllFunc = 106;
char HllDataStr[20];
/* Change session B PS window name */
strcpy(HllDataStr, "B^^^^Monitor");
HllDataStr[4] = 1;
HllDataStr[11] = 0;
int HllDataLgth = 68;
int PsPos;
HLLAPI(&HllFunc, HllDataStr, &HllDataLgth, &PsPos);
```

Appendix A: General troubleshooting procedures

If you have problems running your automation software with Attachmate product, consider the following.

1. **Check that EXTRA! is in the path.** Often the reason an application will fail to start is that the system cannot find the emulator software. At a command prompt, type `EXTRA` and press Enter. A response like "Unknown command or file name" indicates EXTRA! is not in the system search path. Make needed correction, then re-test to verify.

2. **Check the configuration options.** Many problems occur when a session with a short name required by an application has not been configured. Start a session, choose Global Preferences… from the Options menu, then select Advanced properties. Verify that the HLLAPI short name needed by the application has an appropriate session assigned. If not, make needed correction and run the application again to verify.

3. **Check connections.** While faulty cable connections are rare in newer hardware, inspect plugs and jacks to confirm they are securely attached. A more common cause of "failed to connect" errors is improper specification of connection parameters, for example, host TCP/IP network address. Use a technique such as `PING` to check the connection configuration, and correct as necessary.

4. **Check the session.** On occasion, host application programmers may modify content or organization of screens to meet changing need. If workstation automation software has been written to expect specific text in a particular place on a particular screen, software error of some kind is likely to result. Because host applications are rarely changed without notice, systematically review all such advisories. In the event an issue of this type does occur, use a tool such as an API trace to determine exactly where in the software failure occurs, then use that information to identify specifics of the change, and develop appropriate updates for automation software.

5. **Check workload and timings.** If an automation program has been in use for several years, chances are good that hardware at the host, in the network, or the workstation will have been upgraded – or, if not, that workloads on the hardware have changed. In either case, time required to receive and process requests will change, possibly enough that host applications and automation software can get "out of synch", expecting (and trying to process) information that has not yet arrived. Problems like these can be perplexing to diagnose and resolve. Review automation-software logic to verify that suitably robust techniques are being used to synchronize host and workstation operations. If necessary, Attachmate Technical Support can assist by analyzing communications traces to provide information about turnaround times and other details of host/workstation data exchanges.

Appendix B: Host keyboard mnemonics

Table B-1 shows the key codes that allow you to represent special function keys in your calling data strings. You can use these codes with Function 3, “Send Key,” to specify the keystrokes you want to send, as well as with Function 51, “Get Key,” which receives the keystrokes sent through Function 3.

These codes rely on ASCII characters to represent the special function keys of the 3270-PC. For example, to send the keystroke PF1, you would code “@1”. And to represent a System Request keystroke, you would code “@A@H”.

Each key code represents the actual key that is being sent or received. Keep in mind that placing an Alt (@A) or Shift (@S) before a key code will change its meaning. When sending text keystrokes, be sure the codes are entered just as you want them to be received, including the correct case.

Since the Escape character defaults to the at sign (@), you must code the character twice in order to send the escape character as a keystroke. For example, to send a single “@”, you must code “@@”. When your program calls Function 51, “Get Key,” you send a pointer to a keystroke structure used for the returning keystroke. Each keystroke is represented by the following key codes:

• Each key has a number between 1 and 133, which represents the key position on the keyboard.

• Every key has four states: Lower Case, Upper Case, Alt State, and Ctrl State.

Symbols used throughout the tables have the following meanings:

#	Shift keys: this symbol indicates that what follows will be a mnemonic key code.
*	These key positions are not used.
E	A host session’s short name.

Table B-1. Windows keyboard mnemonics

Host key	Mnemonic	Host key	Mnemonic
@	@@	Home	@0
Alternate Cursor	@$	Insert	@I
Attention	@A@Q	Jump	@J
Backspace	@<	New Line	@N
Backtab	@B	Num Lock	@t
Blue	@A@h	Page Down	@v
Caps Lock	@Y	Page Up	@u
Clear	@C	PA1	@x
Cursor Down	@V	PA2	@y
Cursor Left	@L	PA3	@z
Cursor Left Double	@A@L	PF1	@1
Cursor Right	@Z	PF2	@2
Cursor Right Double	@A@Z	PF3	@3
Cursor Select	@A@J	PF4	@4
Cursor Up	@U	PF5	@5
Delete	@D	PF6	@6
Delete Word	@A@D	PF7	@7
Device Cancel	@A@R	PF8	@8
DUP	@S@x	PF9	@9
End	@q	PF10	@a
Enter	@E	PF11	@b
Erase to EOF	@F	PF12	@c
Erase Input	@A@F	PF13	@d
Reset Reverse Video	@A@c	PF14	@e
Field Mark	@S@y	PF15	@f
Green	@A@f	PF16	@g
Reset Host Colors	@A@l	PF17	@h
Reverse Video On	@A@9	PF18	@i
Scr Lock	@s	PF19	@j
System Request	@A@H	PF20	@k
Tab	@T	PF21	@l
Test	@A@C	PF22	@m
Turquoise	@A@i	PF23	@n
Underscore	@A@b	PF24	@o
White	@A@j	Pink	@A@e
Word Tab Back	@A@z	Print PS	@A@T
Word Tab Forward	@A@y	Print Screen	@P
Yellow	@A@g	Queue Overrun	@/
(reserved)	@X	Red	@A@d
Reset	@R	Field Exit	@A@E
Cursor Up Double	@A@U	Cursor Down Double	@A@V

Appendix C: Interpreting the Returned Data String for Function 13

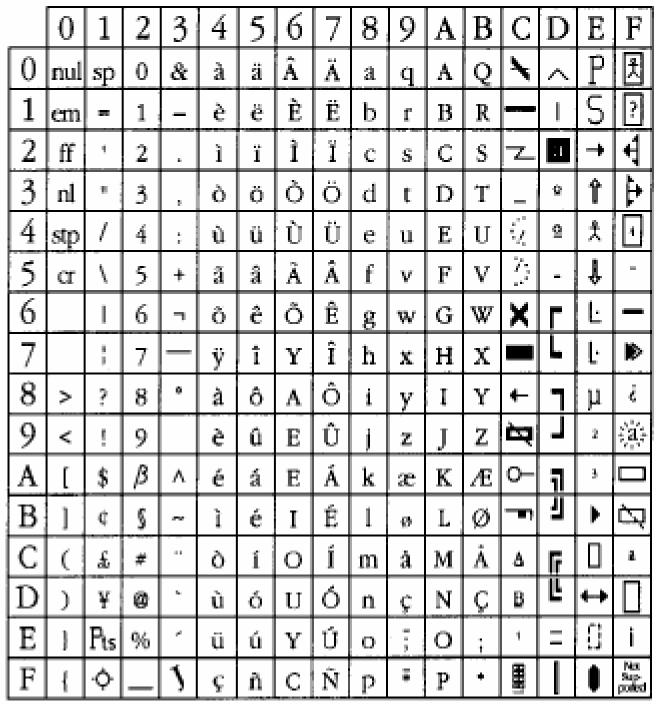
This appendix explains how to decode the data string that Function 13, “Copy OIA,” returns. To interpret this information, you must be able to decipher the OIA image symbols that are returned in positions 2 to 81 of the string, as well as the bits that are returned in positions 82 to 103 of the string.

Position 1 (OIA format byte)

Position 1 of the returning data string always returns the format byte, 1 for 3270 terminal emulation or 9 for 5250.

Positions 2 to 81 (OIA image symbols)

The following chart displays symbols found in the DFT host and CUT host presentation spaces. These symbols can be part of the OIA image returned in positions 2 to 81 of the returning data string.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	nul	sp	0	&	à	ä	Â	Ä	a	q	A	Q	╲	^	P	웃
1	em	=	1	–	è	ë	È	Ë	b	r	B	R	━	|	S	⍰
2	ff	'	2	.	ì	ï	Ì	Ï	c	s	C	S	Z	■	→	⇤
3	nl	"	3	,	ò	ö	Ò	Ö	d	t	D	T	_	°	⇑	⇥
4	stp	/	4	:	ù	ü	Ù	Ü	e	u	E	U	⁘	º	웃	⎕
5	cr	\	5	+	ã	â	Ã	Â	f	v	F	V	⁙	-	⇓	-
6		|	6	¬	õ	ê	Õ	Ê	g	w	G	W	✖	┏	Ŀ	━
7		¦	7	—	ÿ	î	Y	Î	h	x	H	X	▬	┗	Ŀ	▶
8	>	?	8	°	à	ô	A	Ô	i	y	I	Y	←	┓	µ	¿
9	<	!	9		è	û	E	Û	j	z	J	Z	⌧	┛	²	ⓐ
A	[	$	β	^	é	á	E	Á	k	æ	K	Æ	⊸	╗	³	▭
B	]	¢	§	~	ì	é	I	É	l	ø	L	Ø	╕	╝	▶	⌧
C	(	£	#	"	ò	í	O	Í	m	å	M	Å	▵	╔	▯	ª
D	)	¥	@	`	ù	ó	U	Ó	n	ç	N	Ç	ᴮ	╚	↔	▯
E	|	₧	%	´	ü	ú	Y	Ú	o	⁝	O	;	¹	=	⬚	i
F	¦	⌖	_	⧵	ç	ñ	C	Ñ	p	≡	P	•	▤	┃	⬮	Not Sup-ported

Positions 82 to 103 (OIA bit groups)

Remaining positions in the returned data string can be interpreted with the help of the following sections. Each position or group returns a bit number that explains a particular OIA characteristic. The list below summarizes the different groups, the OIA characteristic, and the position number associated with it.

Group	Characteristic explained	Position number
1	Online and Screen Ownership	82
2	Character Selection	83
3	Shift State	84
4	PSS, Part 1	85
5	Highlight, Part 1	86
6	Color, Part 1	87
7	Insert	88
8	Input Inhibited (5 bytes)	89–93
9	PSS, Part 2	94
10	Highlight, Part 2	95
11	Color, Part 2	96
12	Communication Error Reminder	97
13	Printer Status	98
14	Reserved (3270) / Graphic (5250)	99
15	Reserved Group	100
16	Automatic Key Play/Record Status	101
17	Automatic Key Quit/Stop State	102
18	Enlarge State Position	103

Group1: Online and screen ownership

This bit group is the 82nd byte of the OIA data returned to an application by Function 13. This group contains 1 byte of information, describing who owns the current session.

Bit	3270 Description	5250 Description
0	Setup	Reserved
1	Test	Reserved
2	SSCP–LU session owns screen	Reserved
3	LU–LU session owns screen	System available
4	Online and not owned	Reserved
5	Subsystem ready	Subsystem ready
6–7	Reserved	Reserved

Group 2: Character selection

This group is the 83rd byte in the OIA data returned to an application by Function 13. The group contains 1 byte of data and defines the character set currently used in the OIA.

Bit	3270 Description	5250 Description
0	Extended select	Reserved
1	APL	Reserved
2	Kana	Katakana (Japan only)
3	Alphanumeric	Alphanumeric
4	Text	Reserved
5	Reserved	Reserved
6	Reserved	Hiragana (Japan only)
7	Reserved	Double-byte character

Group 3: Shift state

This group is the 84th byte in the OIA data, showing whether caps lock and numeric lock are active.

Bit	3270 Description	5250 Description
0	Upper Shift	Reserved
1	Numeric	Keyboard shift
2	CAPS	CAPS
3-6	Reserved	
7	Reserved	Double-byte char available

Group 4: Program symbol support, part 1

This group is the 85th byte in the OIA data.

Bit	Description
0–7	Reserved

Group 5: Highlight, part 1

This group is the 86th byte in the OIA data and contains highlighting information for the current PS.

Bit	3270 Description	5250 Description
0	User selectable	Reserved
1	Field inherit	Reserved
2–7	Reserved	Reserved

Group 6: Color, part 1

This group is the 87th byte in the OIA data, defining some of the color characteristics being used in the current PS by this operator.

Bit	3270 Description	5250 Description
0	User selectable	Reserved
1	Field inherit	Reserved
2–7	Reserved	Reserved

Group 7: Insert

This group is the 88th byte in the OIA data, defining whether the current PS is in insert mode.

Bit	Description
0	Insert mode
1–7	Reserved

Group 8: Input inhibited

This group consists of bytes 89 through 93 in the OIA data, and indicates why input is inhibited in the current PS.

Byte	Bit	3270 Description	5250 Description
1	0	Non-resettable machine check	Reserved
	1	Reserved for security key	Reserved
	2	Machine check	Reserved
	3	Communications check	Reserved
	4	Program check	Reserved
	5-7	Reserved	Reserved
2	0	Device busy	Reserved
	1	Terminal wait	Reserved
	2	Minus symbol	Reserved
	3	Minus function	Reserved
	4	Too much entered	Reserved
	5-7	Reserved	Reserved
3	0-2	Reserved	Reserved
	3	Invalid dead key combination	Reserved
	4	Wrong place	Reserved
	5	Reserved	Operator input error
	6-7	Reserved	Reserved
4	0-1	Reserved	Reserved
	2	System wait	System wait
	3-7	Reserved	Reserved
5	0-7	Reserved	Reserved

Group 9: Program symbol support, part 2

This is the 94th byte of the OIA data, providing additional information about program symbol support.

Bit	Description
0–7	Reserved

Group 10: Highlight, part 2

This is the 95th byte in the OIA data, and defines more highlight options in the current PS.

Bit	Description
0–7	Reserved

Group 11: Color, part 2

This is the 96th byte in the OIA data. The group defines more color options available to the operator in the information area.

Bit	Description
0–7	Reserved

Group 12: Communications error reminder

This is the 97th byte in the OIA data. Bits in this group define whether the host and the current PS are communicating.

Bit	3270 Description	5250 Description
0	Communications error	Reserved
1–6	Reserved	Reserved
7	Reserved	Message wait

Group 13: Printer status error reminder

This is the 98th byte in the OIA data. Bits in this group describe the status of the printer connected to the current PS.

Bit	Description
0–7	Reserved

Group 14: Reserved (3270) / Graphics (5250)

This is the 99th byte in the OIA data.

Bit	Description
0–7	Reserved

Group 15: Reserved

This is the 100th byte in the OIA data.

Bit	Description
0–7	Reserved

Group 16: Automatic key play/record state

This group is the 101st byte in the OIA data.

Bit	Description
0–7	Reserved

Group 17: Automatic key quit/stop state

This group is the 102nd byte in the OIA data.

Bit	Description
0–7	Reserved

Group 18: Expanded state

This is the 103rd byte in the OIA data.

Bit	Description
0–7	Reserved

Appendix D: Extended Attributes

Function 5, “Copy Presentation Space,” Function 8, “Copy Presentation Space to String,” Function 15, “Copy String to Presentation Space,” Function 33, “Copy String to Field,” and Function 34, “Copy Field to String,” allow an application to access extended attribute bytes (EABs) in a 3270 or 5250 presentation space. Information in this Appendix explains format and interpretation of EABs.

3270 Character Attributes

When a subject function is executed with session parameters EAB and NOXLATE in effect, EAB data are passed to or from a 3270 presentation space in the following format:

Bit	Meaning
0–1	Character highlighting 00 = Normal 01 = Blink 10 = Reverse video 11 = Underline
2-4	Character color 000 = Default 001 = Blue 010 = Red 011 = Pink 100 = Green 101 = Turquoise 110 = Yellow 111 = White
5-7	Reserved

5250 Character Attributes

When a subject function is executed with session parameters EAB and NOXLATE in effect, EAB data are passed to or from a 5250 presentation space in the following format:

Bit	Meaning
0	0 = normal image, 1 = reverse image
1	0 = no underline, 1 = underline
2	0 = no blink, 1 = blink
3	0 = no column separator, 1 = column separator
4-7	Reserved

Color Attributes

When a subject function is executed with session parameters EAB and XLATE in effect, EAB data are translated in the following format to or from application store:

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