SunLink™ SNA 3270 9.1 EHLLAPI Programmer's Manual

The Network Is the Computer™

Sun Microsystems Computer Company
2550 Garcia Avenue
Mountain View, CA 94043 USA
415 960-1300 fax 415 969-9131

Part No.: 802-2668-12
Revision A, August 1997

Contents

Tables

Preface

The SunLink 3270 Extended High-Level Language Application Program Interface (EHLLAPI) lets you build UNIX® applications that interact with IBM host sessions. With EHLLAPI, you can write programs to automate, simplify, and integrate tasks usually done by typing at 3270 display terminals.

Who Should Use This Book

This book documents the use of EHLLAPI program development. Developers should use this book as a guide and reference to the use and features of EHLLAPI. Program developers are expected to be familiar with the 3270 display terminal, its architecture, protocols, and terminology.

How This Book Is Organized

This manual is divided into several chapters, as follows:

Chapter 1, "Introduction to EHLLAPI," provides an overview of the functions and features of EHLLAPI and illustrates how applications written using the API communicate with an IBM host application.

Chapter 2, "Getting Started," introduces you to EHLLAPI server operations and application programming. The chapter guides you through the steps necessary to achieve successful communication between sample programs operating on networked systems. A sample configuration is provided along with step-by-step instructions.

Chapter 3, “Using EHLLAPI,” describes the nature of the EHLLAPI and how it is used to create application programs. This chapter shows you how to organize your program and, with examples, illustrates how the EHLLAPI is used to perform most of the standard 3270 operations.

Chapter 4, “EHLLAPI Functions,” contains detailed specification of the EHLLAPI functions. Each category of functions is introduced and described.

Appendix A, “Sample Programs,” contains three EHLLAPI programs: `query`, `record`, and `playback`.

Appendix B, “Include File,” contains the `ehllapi.h` include file.

Related Documentation

Refer to the following documents for supporting information:

Sun Documentation

- *SunLink IBM 3270 9.0 End Node Planning and Installation Manual*, 802-2665
- *SunLink SNA PU2.1 9.0 Server Configuration and Administration Manual*, 802-2673
- *SunLink Client 3270 9.0 Configuration and User's Guide*, 802-2667

IBM Documentation

- *IBM Personal Communications/3270 Version 2.0*
- *IBM EHLLAPI/DDE Programming Reference*

Typographic Conventions

The following table describes the typographic changes used in this book.

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output.	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`
`AaBbCc123`	What you type, when contrasted with on-screen computer output.	`% su` `Password:`
AaBbCc123	Command-line variable: replace with a real name or value.	To delete a file, type `rm` *filename.*
	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the *User's Guide.* These are called *class* options. You *must* be root to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

Table P-1 Shell Prompts

Shell	Prompt
C shell	`machine_name%`
C shell superuser	`machine_name#`
Bourne shell and Korn shell	`$`
Bourne shell and Korn shell superuser	#

Ordering Sun Documents

SunDocs[SM] is a distribution program for Sun Microsystems technical documentation. Easy, convenient ordering and quick delivery is available from SunExpress. You can find a full listing of available documentation on the World Wide Web: `http://www.sun.com/sunexpress/`

Table P-2 SunExpress Contact Information

Country	Telephone	Fax
United States	1-800-873-7869	1-800-944-0661
United Kingdom	0800-89-88-88	0800-89-88-87
Canada	1-800-873-7869	1-800-944-0661
France	0800-90-61-57	0800-90-61-58
Belgium	02-720-09-09	02-725-88-50
Luxembourg	32-2-720-09-09	32-2-725-88-50
Germany	01-30-81-61-91	01-30-81-61-92
The Netherlands	06-022-34-45	06-022-34-46
Sweden	020-79-57-26	020-79-57-27
Switzerland	0800-55-19-26	0800-55-19-27
Holland	06-022-34-45	06-022-34-46
Japan	0120-33-9096	0120-33-9097

Sun Welcomes Your Comments

Please use the *Reader Comment Card* that accompanies this document. We are interested in improving our documentation and welcome your comments and suggestions.

If a card is not available, you can email or fax your comments to us. Please include the part number of your document in the subject line of your email or fax message.

- Email: `smcc-docs@sun.com`
- Fax: SMCC Document Feedback
  1-415-786-6443

Introduction to EHLLAPI 1

This chapter contains introductory information about the SunLink 3270 Extended High-Level Language Application Program Interface (EHLLAPI) that allows you to build Unix applications that interact with IBM host sessions. It contains information on EHLLAPI program tasks, features, and functions.

1.1 EHLLAPI Program Tasks

Here is a summary of the program tasks associated with the EHLLAPI program.

Task Performance

This task allows an EHLLAPI program to execute repetitive procedures such as automatic logon or job submission.

Keystroke Expansion

This task allows 3270 operators to execute tasks by striking one key. An EHLLAPI program can capture the keystroke and take the appropriate actions.

New User Interface

An EHLLAPI program can be used to interact with one or more IBM host applications and present the consolidated data in a more user-friendly fashion.

Unattended Operation

An EHLLAPI program can be used as a programmed operator. As data is presented by IBM host applications, your program can recognize important messages and react by entering the appropriate information.

Distributed Processing

You can handle some jobs on your Unix system and use IBM host applications for other tasks (for example, database searches, and updates).

SunLink EHLLAPI is an important tool for developers who want to integrate the IBM mainframe and Unix workstation environments. Almost all IBM host applications provide a 3270 display terminal interface. The SunLink EHLLAPI can be used to control 3270 sessions with Unix applications.

The EHLLAPI parallels the IBM Personal Communications/3270 High-Level Language Application Program Interface. Refer to the *IBM EHLLAPI/DDE Programming Reference Manual.*

1.2 EHLLAPI Functions and Features

SunLink EHLLAPI provides a programming interface to IBM host applications that usually interact with 3270 display terminals. EHLLAPI provides you with access to the display buffer and keyboard of up to eight emulated 3270 display terminals.

Your EHLLAPI programs can run on Unix workstations throughout your local network environment. The EHLLAPI server uses TCP/IP services to attach to the Sun SNA server that supports the IBM host sessions.

Table 1-1 lists all the EHLLAPI functions and indicates which functions are available to SunLink EHLLAPI users (all functions, except Storage Manager [17], which is an obsolete function, are supported).

Table 1-1 EHLLAPI Functions

Function Number	Function Name	Sun Support
1	Connect Presentation Space	Yes
2	Disconnect Presentation Space	Yes
3	Send Key	Yes
4	Wait	Yes
5	Copy Presentation Space	Yes
6	Search Presentation Space	Yes
7	Query Cursor Location	Yes
8	Copy Presentation Space to String	Yes
9	Set Session Parameters	Yes
10	Query Sessions	Yes
11	Reserve	Yes
12	Release	Yes
13	Copy OIA	Yes
14	Query Field Attribute	Yes
15	Copy String to Presentation Space	Yes
17	Storage Manager	No (obsolete)
18	Pause	Yes
20	Query System	Yes
21	Reset System	Yes
22	Query Session Status	Yes
23	Start Host Notification	Yes
24	Query Host Update	Yes
25	Stop Host Notification	Yes
30	Search Field	Yes
31	Find Field Position	Yes
32	Find Field Length	Yes

Table 1-1 EHLLAPI Functions *(Continued)*

Function Number	Function Name	Sun Support
33	`Copy String to Field`	Yes
34	`Copy Field to String`	Yes
40	`Set Cursor`	Yes
50	`Start Keystroke Intercept`	Yes
51	`Get Key`	Yes
52	`Post Intercept Status`	Yes
53	`Stop Keystroke Intercept`	Yes
90	`Send File`	Yes
91	`Receive File`	Yes
99	`Convert Position or RowCol`	Yes

1.3 EHLLAPI Components

SunLink EHLLAPI contains several components that are listed in Table 1-2.

Table 1-2 EHLLAPI Components

Component	Function
`ehllapi.o`	A compile time object module linked with your program
`ehllapi.h`	Header files defining the EHLLAPI interface
`sun3270x, sun3270tty`	3270 display terminal emul ator
`sunpu2.1`	Sun SNA server that provides connectivity to IBM hosts

Appendix A, “Sample Programs,” contains sample programs illustrating basic EHLLAPI function calls.

Getting Started 2

This chapter guides you through the steps necessary to establish communication between an EHLLAPI program and an IBM host application. Sun distributes three sample EHLLAPI programs with the `SUNWopcl` package (in the `prg_samples` directory). The programs are listed in Table 2-1.

Table 2-1 EHLLAPI Sample Programs

Program	Function
`query`	Displays position and attribute information of the cursor location in the 3270 display buffer.
`record`	Records keystrokes you enter into the 3270 session and attempts to remember when and where keystrokes were entered.
`playback`	Uses the information captured by `record` to replay the previous actions you took when interacting with IBM host applications via the 3270 session.

These programs are compiled and linked with the EHLLAPI library `ehllapi.o`.

Before you can run the programs you must install, configure, and activate the SunLink PU2.1 Gateway and the SunLink client 3270 (the EHLLAPI server). These steps are generally the responsibility of the Unix system administrator. Sun delivers SunLink client 3270 with installation scripts and sample configurations to simplify the process.

2

The remaining sections of this chapter describe how to start and use the sample programs. You need to become familiar with the SunLink EHLLAPI programming environment by trying to use and understand these programs.

2.1 SunLink PU2.1 SNA Server Startup

The SunLink PU2.1 SNA server provides the underlying connectivity to the IBM SNA network, which EHLLAPI programs use to communicate with IBM host applications. The SunLink PU2.1 SNA server software must be installed, configured, and started before an EHLLAPI program can communicate with an IBM host application.

Refer to the "Getting Started" chapter in the *SunLink SNA PU2.1 9.0 Server Configuration and Administration Manual* for step-by-step instructions on how to configure and start the SunLink SNA server.

2.2 SunLink 3270 Startup

The SunLink client 3270 9.0 product can act as an EHLLAPI server. EHLLAPI programs attach to a SunLink 3270 application to access the associated 3270 keyboard and display buffer. You need to indicate that a SunLink 3270 application supports EHLLAPI programs by associating it with a well-known EHLLAPI session name. Well-known EHLLAPI session names are only one character (A-Z, a-z, 0-9). Or you can have the operating system assign a transient port. To conform to the EHLLAPI standard, see the *EHLLAPI/DDE Programming Reference*. Only session names A-H should be used. There are two methods for creating this association:

1. Specify the EHLLAPI session name when invoking the SunLink 3270 client application. For example:

   ```
   sun3270x -e A
   ```

   links the EHLLAPI session named "A" with this `sun3270x` invocation.

2. Configure the EHLLAPI session name in the SunLink SNA PU2.1 9.0 server configuration for an LU with `lutype = 2` (3270 display terminal). For example:

```
LU      name=HLA_A,
        lutype=2,
        locaddr=2;
```

   where the first three characters, “HLA,” indicate that this LU can be an EHLLAPI server and the last character, “A,” specifies the EHLLAPI session name.

Both of these methods require that the service name `sunehllapiA` be defined in the `/etc/services` file (NIS services database). EHLLAPI applications connect via the well-known EHLLAPI session short name. If the SunLink 3270 client application is started without the `-e` command-line option, the operating system assigns a transient port for EHLLAPI connections. In this case, no entry is required in the `/etc/services` file (NIS services database). However, EHLLAPI applications must learn the transient port number before a connection to the SunLink 3270 client application can be realized. This is primarily intended to facilitate file transfer without the need for `/etc/services` file (NIS services database) updates or the `-e` command line option. The systems-defined transient port number *may* be communicated to any application via the SunLink 3270 client user exits and user-defined function keys.

Refer to the “Getting Started” section in *SunLink Client 3270 9.0 Configuration and User's Guide* for step-by-step instructions on how to configure and start `sun3270x`.

2.3 EHLLAPI Program Linking

To link the sample EHLLAPI programs, execute `make` in the `prg_samples` directory. The `Makefile` in this directory specifies the compilation and link commands for the sample EHLLAPI programs.

To compile and link a small EHLLAPI program called test, the command line will look similar to:

```
cc -o test test.c ehllapi.o
```

See the man pages for `cc` for additional compilation and link options.

2.4 EHLLAPI Program Execution

You can call the sample programs while the `sun3270x` application is running. All the sample programs connect to the EHLLAPI session named "`A`." Therefore, the `sun3270x` application should be started with the "`-e A`" option.

2.5 EHLLAPI Sample Programs

There are three EHLLAPI programs that are described in the subsections that follow.

2.5.1 `query`

Start the `query` program (there are no command options). Type `i` followed by Return at the prompt. The current position of the cursor is displayed along with the character attributes for that position. Move the cursor to another position in the `sun3270x` application window. Again, type `i` followed by Return at the prompt to display the cursor position and character attributes for the new position.

2.5.2 `record`

Start `record` when you want to begin logging keystrokes in the `sun3270x` window. The `record` program writes the keystrokes with their screen context to the session file. Type Control-C to exit the `record` program.

2.5.3 `playback`

Start `playback` when you have reestablished the `sun3270x` window contents to match the beginning of the `record` session. The `playback` program replays the keystrokes that `record` captured earlier and stored in the session file. The `playback` program will enter keystrokes immediately if it recognizes the screen context. Otherwise, `playback` pauses 15 seconds before entering the next keystroke.

Using EHLLAPI 3

This chapter outlines programming constructs for using EHLLAPI to control 3270 presentation spaces. The following topics are included:

- Presentation space interactions
- EHLLAPI function call format
- EHLLAPI function descriptions
- ASCII mnemonics for keystroke capture and entry
- Linking EHLLAPI with an application

3.1 An Approach to EHLLAPI Programming

Before implementing an EHLLAPI program, you need to thoroughly understand the sequence of events that the host expects while performing a given task. Think of EHLLAPI programs as “programmed operators.”

You should explicitly list how to recognize important screen images and when to enter keystrokes. Practice the task using SunLink 3270 client, and then try programming the task with EHLLAPI.

Timing is critical. An EHLLAPI program can enter keystrokes and examine data much faster than an operator. Moreover, the IBM host application may send information more slowly if the IBM host is busy or the network is congested. After sending keystrokes, you should use the `Wait` function to determine when the host has responded to your request. Use the search

functions to recognize when the host application displays expected screens. Only enter keystrokes or data after the IBM host application displays the expected screen.

Check return codes for important EHLLAPI functions (in particular, `Wait`, `Pause`, `Send Key`). Make sure that your session is in the proper state. Otherwise, you cannot perform the task you would like to perform.

Become familiar with the EHLLAPI "behavioral" parameters, which are controlled by using the `Set Session Parameters` function. The `AUTORESET` parameter prefixes all keystrokes sent by the EHLLAPI program with the Reset key (the Reset key resets the 3270 display just in case input is inadvertently inhibited). The `IPAUSE` parameter enables your program to use the `Pause` function to wait for presentation space and/or operator interface (OIA) updates.

3.2 EHLLAPI Functions

EHLLAPI provides functions to interact with the 3270 display buffer and keyboard. There are several types of functions available. See Table 3-1.

Table 3-1 EHLLAPI Function Types

Function Type	Usage
Connection	Associate your EHLLAPI program with a 3270 presentation space.
Session Control	Set parameters that control the behavior of EHLLAPI functions. For example, searches can search forward or backward, and copies can include attribute bytes or you can replace them with space characters.
Keystroke Handling	Enable your EHLLAPI program to send keystrokes to the 3270 presentation space or to intercept keystrokes entered by a 3270 operator.
Copy	Copy data to/from the 3270 presentation spaces.
Search	Search for data in the 3270 presentation space.
Notification	Inform your program when the status of the 3270 presentation space or Operator Interface Area (OIA) changes.
Information	Query EHLLAPI about the characteristics of the 3270 presentation space.
File Transfer	Facilitate transfer of files to/from the IBM host.

Table 3-2 lists EHLLAPI functions grouped according to function types.

Table 3-2 EHLLAPI Functions Grouped by Function Types

Function Type	Function	Sun Support
Connection Functions		
	`Connect Presentation Space (1)`	Connect to specified PS.
	`Disconnect Presentation Space (2)`	Disconnect from current connected PS.
Session Control Functions		
	`Set Session Parameters (9)`	Change behavior of EHLLAPI function(s).
	`Reset System (21)`	Start EHLLAPI from the default state.
Keystroke Handling Functions		
	`Send Key (3)`	Send keystrokes to current connected PS.
	`Reserve (11)`	Stop user input in current connected PS.
	`Release (12)`	Re-allow user input in current connected PS.
	`Set Cursor (40)`	Position cursor in current connected PS.
	`Start Keystroke Intercept (50)`	Begin capturing keystrokes from a PS.
	`Get Key (51)`	Get keystroke from a PS.
	`Post Intercept Status (52)`	Accept or reject intercepted keystroke.
	`Stop Keystroke Intercept (53)`	End keystroke interception for a PS.
Copy Functions		
	`Copy Presentation Space (5)`	Copy entire connected PS to program buffer.
	`Copy Presentation Space to String (8)`	Copy part of connected PS to program buffer.
	`Copy OIA (13)`	Copy Operator Interface Area to program buffer.
	`Copy String to Presentation Space (15)`	Copy string into connect PS.
	`Copy String to Field (33)`	Copy string to field in connected PS.
	`Copy Field to String (34)`	Copy field in connected PS to program buffer.
Search Functions		
	`Search Presentation Space (6)`	Search connected PS for string.
	`Search Field (30)`	Search field in connected PS for string.
	`Find Field Position (31)`	Find position of field with given attributes.
	`Find Field Length (32)`	Find length of field with given attributes.

Table 3-2 EHLLAPI Functions Grouped by Function Types *(Continued)*

Function Type	Function	Sun Support
Notification Functions		
	`Wait (4)`	Wait for host response to connected PS.
	`Pause (18)`	Sleep or wait for host to update PS or OIA.
	`Start Host Notification (23)`	Request notifications on PS or OIA update.
	`Query Host Update (24)`	Check type of notification received.
	`Stop Host Notification (25)`	End PS or OIA update notifications.
Information Functions		
	`Query Cursor Location (7)`	Determine position of cursor in connected PS.
	`Query Sessions (10)`	Determine PS size for all host sessions.
	`Query Field Attribute (14)`	Determine attributes for field in connected PS.
	`Query System (20)`	Determine system-related values.
	`Query Session Status (22)`	Determine characteristics of a PS.
	`Convert Position or RowCol (99)`	Determine PS position from/to row/column.
File Transfer Functions		
	`Send File (90)`	Send a file to the host.
	`Receive File (91)`	Receive a file from the host.

3.3 Presentation Space Connections

EHLLAPI facilitates the interaction between your program and one or more 3270 presentation spaces. The presentation space name associated with a particular SunLink 3270 client emulation is specified when the SunLink 3270 client is invoked. The presentation space name may be a well-known value or an operating system assigned transient port. Well-known values are defined as the letters “a” through “z,” “A” through “Z,” and the digits 0 through 9. Well-known values *must* have an associated service specified in the `/etc/services` (NIS services database). Transient ports are used when a well-known value is not specified at SunLink 3270 client invocation. In this case, the SunLink 3270 client requests the operating system to assign a port. No entry in the `/etc/services` (NIS services database) is required. Applications

connecting to a presentation space defined by a transient port must obtain the port number following SunLink 3270 client initiation (refer to `actCommand` and `userCommand` options of the SunLink 3270 client).

A presentation space can range in size from 1920 bytes (24 rows by 80 columns) to 3564 bytes (27 rows by 132 columns).

The `Connect Presentation Space` function connects a program to a presentation space. Most EHLLAPI functions only interact with connected presentations spaces. An EHLLAPI application can only be connected to one 3270 presentation. The last presentation space identified in a `Connect Presentation Space` function is the current connected presentation space (a `Disconnect Presentation Space` does not have to be issued before connecting to a different presentation space).

Other EHLLAPI functions identify the target presentation space by placing the short presentation name in the first character of a data string. When the presentation space is identified by a transient port, the first character of the data string *must* be a space (indicating the currently connected presentation space).

3.4 Function Parameters

All EHLLAPI function calls expect four parameters. Depending upon the value of the function parameter, the other parameters can take several different meanings. The EHLLAPI calling syntax is provided in Code Example 3-1.

Code Example 3-1 EHLLAPI Calling Syntax

```
hllc(function, data_string, length, return_code)
         int *function;
         char *data_string;
         int *length;
         int *return_code;
```

In general, the following rules apply:

- `function` specifies the EHLLAPI function. These functions are described in detail in the following chapter.
- `data_string` specifies space to share data between EHLLAPI and the program.
- `length` specifies the size of the data string. For some functions the length value can be overridden by using the session option `STREOT` (see `Set Session Parameters` in Table 3-2).
- `return_code` can specify a position in the presentation space on input. The status of the function call is stored on output.

Note that the calling parameters change their meaning depending upon the function call. Refer to each function's description to determine its correct parameter setup.

3.5 ASCII Mnemonics

Keystrokes for a 3270 presentation space are represented by the corresponding ASCII value or by an ASCII mnemonic for keystrokes outside the ASCII character data set.

Mnemonic keystrokes are introduced by an escape character (@), which is the default escape character. The `Set Session Parameters` function can change the escape character.

The `Get Key` and the `Send Key` functions use the ASCII mnemonics.

Table 3-3 through Table 3-5 list the ASCII mnemonics for 3270 presentation space keystrokes.

Table 3-3 Single-Character Escape Mnemonics

Mnemonic	Description	Mnemonic	Description
@B	Backtab	@T	Tab
@C	Clear*	@U	Cursor Up
@D	Delete	@V	Cursor Down
@E	Enter*	@Y	Caps Lock†
@F	Erase EOF	@Z	Cursor Right

Table 3-3 Single-Character Escape Mnemonics *(Continued)*

Mnemonic	Description	Mnemonic	Description
@I	Insert	@s	Scroll Lock†
@J	Jump†	@t	Num Lock†
@L	Cursor Left	@@	@ (at) symbol
@N	New Line	@$	Alternate Cursor†
@P	Print	@<	Back Erase
@R	Reset		

*AID key mnemonic.

†Accepted but ignored by EHLLAPI server. Never returned by `Get Key`.

Table 3-4 Function Key Mnemonics

Mnemonic	Description	Mnemonic	Description
@0	Home	@e	PF14*
@1	PF1*	@f	PF15*
@2	PF2*	@g	PF16*
@3	PF3*	@h	PF17*
@4	PF4*	@i	PF18*
@5	PF5*	@j	PF19*
@6	PF6*	@k	PF20*
@7	PF7*	@l	PF21*
@8	PF8*	@m	PF22*
@9	PF9*	@n	PF23*
@a	PF10*	@o	PF24*
@b	PF11*	@x	PA1*
@c	PF12*	@y	PA2*
@d	PF13*	@z	PA3*

*AID key mnemonic.

†Accepted but ignored by EHLLAPI server. Never returned by `Get Key`.

Table 3-5 Attention and Shift Mnemonics

Mnemonic	Description	Mnemonic	Description
@A@F	Erase Input	@A@b	Underscore†
@A@H	System Request*	dq@A@y	Forward Word Tab
@A@J	Cursor Select*†	@A@z	Backward Word Tab
@A@Q	Attention*	@S@x	Dup
@A@R	Device Cancel†	@S@y	Field Mark
@A@T	Print Screen		

*AID key mnemonic.

†Accepted but ignored by EHLLAPI server. Never returned by `Get Key`.

3.6 Return Codes

You should check the return code after all EHLLAPI functions. Table 3-6 lists the return codes and their meanings.

Table 3-6 Return Codes

Code	Name	Explanation
0	`HRC_SUCCESSFUL`	Operation successful or no updates.
1	`HRC_PS_ID_INVALID`	Invalid presentation space specified.
2	`HRC_PARAMETER_ERROR`	Error in specifying parameters or invalid function.
3	`HRC_FT_COMPLETE`	File transfer complete.
4	`HRC_PS_BUSY`	Presentation space is busy.
5	`HRC_FUNCTION_INHIBITED`	Presentation space is locked.
6	`HRC_DATA_ERROR`	Warning. The input parameter length and the field length do not match. The data may be truncated.
7	`HRC_PS_POSITION_INVALID`	The presentation space position is not valid.
8	`HRC_PROCEDURE_ERROR`	Function called out of turn.
9	`HRC_SYSTEM_ERROR`	Unsuccessful; system error encountered.

Table 3-6 Return Codes *(Continued)*

Code	Name	Explanation
10	HRC_FUNCTION_UNAVAILABLE	Unavailable or unknown function.
11	HRC_RESOURCE_UNAVAILABLE	Presentation space is not available; another EHLLAPI application is already connected to the presentation space.
20	HRC_UNDEFINED_COMBINATION	Undefined key combination typed.
21	HRC_OIA_UPDATED	OIA updated.
22	HRC_PS_ONLY_UPDATED	Presentation space updated.
23	HRC_PS_OIA_UPDATED	Both presentation space and OIA updated.
24	HRC_PS_UNFORMATTED	Presentation space is not field-formatted.
25	HRC_KEYSTROKES_UNAVAILABLE	Keystrokes not available on input queue.
26	HRC_PS_UPDATED	Warning. Presentation space updated or OIA updated before pause elapsed.
27	HRC_FT_TERMINATED	File transfer terminated early.
28	HRC_FIELD_ZERO_LENGTH	Field has zero length.
31	HRC_KEYSTROKES_LOST	Keystrokes lost due to input queue overflow.
9998	HRC_PS_ID_INVALID_99	Invalid presentation space specified.
9999	HRC_PARAMETER_ERROR_99	'P' or 'R' is not present in second character of input parameter `data_string`.

3.7 Tracing

EHLLAPI has a trace mechanism. Use tracing to help debug your EHLLAPI program.

To start tracing, issue the `Set Session Parameter` function call and select the `TRON` option. The EHLLAPI trace points are recorded in the `hllc_trace` file (`hllc_trace.1` represents the previous 1000 trace points).

3.8 Linking EHLLAPI Programs

Link EHLLAPI with your program by including `ehllapi.o` in the link directive. For example, to compile and link a small program, enter:

```
cc -o example example.c ehllapi.o
```

EHLLAPI Functions 4

This chapter lists all the SunLink EHLLAPI functions in alphabetical order. A numbered list by function number is presented in Table 1-1. Appendix A, “Sample Programs,” includes samples of the use of a function.

In this chapter, each function is listed by name along with function number.

4.1 Connect Presentation Space (1)

Synopsis

```
#include            “ehllapi.h”
function = HFUN_CONNECT_PS;    /* function = 1             */
strcpy(data_string, “A”);      /* connect to PS named “A”  */
length = 1;                    /* length of PS name        */
hllc (&function, data_string, &length, &return_code);
```

Description

Establishes the connection between the EHLLAPI application and the designated 3270 presentation space.

Prior Calls

None

Input Parameters

Table 4-1 `Connect Presentation Space` Inputs

Input	Input Type	Function
`function`	ptr to integer	Function number (1).
`data_string`	ptr to character string	Short name of 3270 presentation space. This is a NULL terminated ASCII string. For well-known presentation space names, it is one (1) character string containing the presentation space short name (for example, a-z, 0-9, A-Z). For transient ports, this is an ASCII string representing the port number (maximum four digits).
`length`	ptr to integer	Not applicable (value 1 is implicit).

Output Parameters

`Connect Presentation Space` returns a return code that is a pointer to an integer with one of the values listed in Table 4-2 below.

Table 4-2 `Connect Presentation Space return_code` Values

Value	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful connection; presentation space is unlocked and ready for input.
1	`HRC_PS_ID_INVALID`	Unsuccessful connection; specified presentation space does not exist.
4	`HRC_PS_BUSY`	Successful connection; presentation space is busy.
5	`HRC_FUNCTION_INHIBITED`	Successful connection; presentation space is locked.
9	`HRC_SYSTEM_ERROR`	Unsuccessful connection; system error encountered.
11	`HRC_RESOURCE_UNAVAILABLE`	Unsuccessful connection; presentation space is not available—another EHLLAPI application is already connected to the presentation space.

Session Parameters

Not applicable

Comments

The current connected presentation space is the last connected presentation. Therefore, function calls requiring a prior `Connect Presentation Space` use the current connected presentation space.

Functions calls requiring a prior `Connect Presentation Space` are listed in Table 4-3.

Table 4-3 Function Calls Requiring `Connect Presentation Space`

Function 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`
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`

Only one EHLLAPI application can connect to a presentation space at a time.

A presentation space exists only if:

- The presentation space name is defined in `/etc/services` (NIS services database).
- `sun3270` client is attached to the LU port with the presentation space name.

See Also

`Disconnect Presentation Space`

4.2 Convert Position or RowCol (99)

Synopsis

```
#include          “ehllapi.h”
function = HFUN_CONVERT_POSITION_OR_ROWCOL;  /* function = 99     */
strcpy(data_string, “AP”);             /* for presentation space A:
                                          convert position to rowcol*/
return_code = 1;                         /* PS position       */
hllc (&function, data_string, &length, &return_code);
```

Description

Converts a presentation space positional value to a screen display row/column coordinate, or converts a screen display row/column coordinate into a presentation space positional value.

This is the only EHLLAPI function for which the fourth parameter (`return_code`) does not return a normal return code value. Error-handling routines must treat this function differently.

Prior Calls

None

Input Parameters

The input parameters are described in Table 4-4.

Table 4-4 `Convert Position` or `RowCol` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (99).
`data_string`	ptr to character string	Short name of 3270 presentation space (one-character A-Z, a-z, 0-9, or space) and a conversion indicator (one-character P or R). 'P' indicates convert presentation space position to row/column. 'R' indicates convert row/column to presentation space position.
`length`	ptr to integer	Not applicable when 'P' is specified. Row when 'R' is specified. Lower limit for rows is 1.
`return_code`	ptr to integer	Presentation space position when 'P' is specified. Lower limit for position is 1; column when 'R' is specified. Lower limit for columns is 1.

Note – When the presentation space is identified by a transient port, the presentation space name *must* be specified as a space (indicating the currently connected presentation space).

Output Parameters

The output parameters are shown in Table 4-5.

Table 4-5 `Convert Position` or `RowCol Outputs`

Output	Input Type	Function
`length`	ptr to integer	Row when 'P' is specified. The number of the row that contains the presentation space position specified in the input parameter `return_code`. Error when 'R' is specified. A value of 0 indicates an error in the row value specified, in the input parameter `length`.
`return_code`	ptr to integer	Value of 0, 9998, or 9999 are error return codes. Column when 'P' is specified. Then number of the column that contains the presentation space position specified in the input parameter `return_code`. Position when 'R' is specified. The presentation space position corresponding to the row and column values specified in the input parameters `length` and `return_code`.

Supported output `return_code` values are listed in Table 4-6.

Table 4-6 `Convert Position` or `RowCol return_code` Values

Value	Name	Explanation
0	`HRC_BAD_PARAMETER_99`	Incorrect presentation space or column.
>0		Presentation space or column.
9998	`HRC_PS_ID_INVALID_99`	Invalid presentation space specified.
9999	`HRC_PARAMETER_ERROR_99`	P or R is not present in second character of input parameter `data_string`.

Session Parameters

Not applicable

Comments

Use the Query Session Status function call to determine the rows and columns of the presentation space.

See Also

Query Session Status

4.3 Copy Field to String (34)

Synopsis

```
#include           "ehllapi.h"
function = HFUN_COPY_FIELD_TO_STRING;    /* function = 34          */
                                        /* data_string allocated    */
length = 6;                              /* copy 6 bytes to string   */
return_code = 1;                         /* PS position of field     */
hllc (&function, data_string, &length, &return_code);
```

Description

Copies characters from a field in the current connected presentation space into a string provided by the EHLLAPI application. The number of characters to copy is specified in the length parameter. The field is identified by the presentation space position of the field, or the presentation space position of any character in the field, as specified in the return_code parameter. This function begins the copy with the first character in the field. The copy ends when the end of the field is reached, or when the target string is full.

This function can copy characters from protected or unprotected fields, but it can only be used with field-formatted presentation spaces. Characters in the presentation space are converted into ASCII characters by this function—attribute bytes and unrepresented ASCII characters are normally translated into blanks (see Set Session Parameters for other cases).

Prior Calls

Connect Presentation Space

Input Parameters

Table 4-7 `Copy Field to String` Inputs

Input	Input Type	Function
`function`	ptr to integer	Function number (34).
`data_string`	ptr to character string	Preallocated memory to hold the copied string.
`length`	ptr to integer	Maximum number of bytes to copy.
`return_code`	ptr to integer	Presentation space position of any byte in the target field. The copy starts with the first character.

Output Parameters

Table 4-8 `Copy Field to String` Outputs

Output	Output Type	Function
`data_string`	ptr to character string	String containing characters copied from the field identified by the input parameter `return_code` in the connected presentation space. The length of the string is the smaller of: the length specified in the input parameter `length` or the length of the field.
`return_code`	ptr to integer	Status of function call. Return values supported as listed in Table 4-9.

Table 4-9 `Copy Field to String return_code` Values

Code	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful copy; presentation space is unlocked and ready for input.
1	`HRC_PS_ID_INVALID`	Unsuccessful copy; specified presentation space does not exist.
2	`HRC_PARAMETER_ERROR`	Unsuccessful copy; error in specifying parameters.

Table 4-9 Copy Field to String return_code Values *(Continued)*

Code	Name	Explanation
6	HRC_DATA_ERROR	Warning. The input parameter length and the field length do not match. The data may be truncated.
7	HRC_PS_POSITION_INVALID	Unsuccessful copy; the presentation space position is not valid.
9	HRC_SYSTEM_ERROR	Unsuccessful copy; system error encountered.
24	HRC_PS_UNFORMATTED	Unsuccessful copy; presentation space is not field-formatted.

Session Parameters

EAB/NOEAB and XLATE/NOXLATE affect the operation of Copy Field to String.

Comments

To determine field position use the function Find Field Position.
To obtain the length of field use the function Find Field Length.

Memory for the parameter data_string must be preallocated before using this function.

See Also

Connect Presentation Space, Find Field Position,
Find Field Length, Set Session Parameters

4.4 Copy OIA (13)

Synopsis

```
#include          "ehllapi.h"
function = HFUN_COPY_OIA;              /* function = 13          */
                                      /* data_string allocated   */
length = 103;                          /* 103 bytes for OIA info  */
hllc (&function, data_string, &length, &return_code);
```

Description

Returns the Operator Information Area (OIA) information for the current connect presentation space. The OIA is located on the bottom line of the display screen (below the line). It displays session status information about the 3270 connection to the IBM host.

Prior Calls

Function Number (1) `Connect Presentation Space`

Input Parameters

Table 4-10 `Copy OIA` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (13).
`data_string`	ptr to character string	Preallocated memory to hold the copied OIA information.
`length`	ptr to integer	Bytes allocated (103).

Output Parameters

Table 4-11 `Copy OIA` Outputs

Output	Output Type	Usage
`data_string`	ptr to character string	String (103 bytes) containing OIA information. String contents are described in the next section, “OIA Information.”
`return_code`	ptr to integer	Status of function call. Supported `return_code` values are listed in Table 4-12.

Table 4-12 Copy OIA return_code Values

Code	Name	Explanation
0	HRC_SUCCESSFUL	Successful copy; presentation space is unlocked and ready for input.
1	HRC_PS_ID_INVALID	Unsuccessful copy; specified presentation space does not exist.
2	HRC_PARAMETER_ERROR	Unsuccessful copy; error in specifying parameters.
4	HRC_PS_BUSY	Successful copy; presentation space is busy.
5	HRC_FUNCTION_INHIBITED	Successful copy; presentation space is locked.
9	HRC_SYSTEM_ERROR	Unsuccessful copy; system error encountered.

OIA Information

Table 4-13 OIA Information Format

Byte	Bit	Indication
0	0-7	Format byte
1-80	80-byte ASCII string	OIA Image Group representing displayed OIA
Bytes 81-102 OIA Group Indicators		
81 — On-line and Screen Ownership	0-1	Reserved
	2	SSCP-LU session owns screen
	3	LU-LU session owns screen
	4	On-line and not owned
	5	Subsystem ready
	6-7	Reserved
82 — Character Selection		
	0-7	Reserved
83 — Shift State		
	0-7	Reserved
84 — Reserved	0-7	Reserved

Table 4-13 OIA Information Format *(Continued)*

Byte	Bit	Indication
85 — Highlight	0-7	Reserved
86 — Color	0-7	Reserved
87 — Insert	0	Insert Mode
	1-7	Reserved
Bytes 88-92 — Input Inhibited		
88	0	Non-resettable machine check
	1	Reserved
	2	Machine check
	3	Communication check
	4	Program check
	5-7	Reserved
89	0	Device busy
	1, 2	Reserved
	3	Minus function
	4	Too much entered
	5-7	Reserved
90	0-2	Reserved
	3	Invalid dead key combination
	4	Wrong place
	5-7	Reserved
91	0-1	Reserved
	2	System wait
	3-7	Reserved
92	0-7	Reserved
93	0-7	Reserved
94	0-7	Reserved
95	0-7	Reserved
96 — Communication Error	0	Communication error
	1-7	Reserved
97	0-7	Reserved
Byte 98 — Graphics	0	Graphics cursor

Table 4-13 OIA Information Format *(Continued)*

Byte	Bit	Indication
	1-7	Reserved
99	0-7	Reserved
100	0-7	Reserved
101	0-7	Reserved
102	0-7	Reserved

Session Parameters

Not applicable

4.5 Copy Presentation Space (5)

Synopsis

```
#include           "ehllapi.h"
function = HFUN_COPY_PRESENTATION_SPACE;  /* function = 34        */
/* data_string allocated    */
hllc (&function, data_string, &length, &return_code);
```

Description

Copies the contents of the connected presentation space into the `data_string` parameter. Characters in the presentation space are converted into ASCII characters by this function—attribute bytes and unrepresented ASCII characters are normally translated into blanks (see `Set Session Parameters` for other cases). See Table 4-14 for input parameters details.

Prior Calls

Function Number (1) `Connect Presentation Space`

Input Parameters

Table 4-14 Copy Presentation Space Inputs

Input	Input Type	Usage
function	ptr to integer	Function number (5).
data_string	ptr to character string	Preallocated memory to hold the copied string. This target string must be as large as the size of the presentation space.
length	ptr to integer	Not applicable (size of presentation space is implicit).

Output Parameters

Table 4-15 Copy Presentation Space Outputs

Output	Output Type	Usage
data_string	ptr to character string	String containing characters copied from the connected presentation space.
return_code	ptr to integer	Status of function call. Supported return_code values are listed in Table 4-16.

Table 4-16 Copy Presentation Space return_code Values

Value	Name	Explanation
0	HRC_SUCCESSFUL	Successful copy; presentation space is unlocked and ready for input.
1	HRC_PS_ID_INVALID	Unsuccessful copy; specified presentation space does not exist.
2	HRC_PARAMETER_ERROR	Unsuccessful copy; error in specifying parameters.
4	HRC_PS_BUSY	Successful copy; presentation space is busy.
5	HRC_FUNCTION_INHIBITED	Successful copy; presentation space is locked.
9	HRC_SYSTEM_ERROR	Unsuccessful copy; system error encountered.

Session Parameters

ATTR/NOATTR, EAB/NOEAB, XLATE/NOXLATE affect the operation of Copy Presentation Space.

Comments

To copy a portion of the presentation space, use Copy Presentation Space to String function.

Memory for the parameter data_string must be preallocated before using this function.

See Also

Connect Presentation Space, Copy Presentation Space to String, Set Session Parameters

4.6 copy Presentation Space to String (8)

Synopsis

```
#include          "ehllapi.h"

function = HFUN_COPY_PS_TO_STRING;     /* function = 8          */
                                      /* data_string allocated  */
length = 6;                           /* copy 6 bytes to string */
return_code = 1;                      /* PS position copy start */
hllc (&function, data_string, &length, &return_code);
```

Description

Copies all or part of the current connected presentation space into a string provided by the EHLLAPI application. The number of characters to copy is specified in the length parameter. The copy begins at the presentation space position identified by the return_code parameter. The presentation space condition plus the copy length cannot exceed the size of the presentation space.

Characters in the presentation space are converted into ASCII characters by this function—attribute bytes and unrepresented ASCII characters are normally translated into blanks (see `Set Session Parameters` for other cases).

Prior Calls

Function Number (1) `Connect Presentation Space`

Input Parameters

Table 4-17 `Copy Presentation Space to String` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (8).
`data_string`	ptr to character string	Preallocated memory to hold the copied string.
`length`	ptr to integer	Number of bytes to copy.
`return_code`	ptr to integer	Presentation space position at which to begin copy.

Output Parameters

Table 4-18 `Copy Presentation Space to String` Outputs

Output	Output Type	Usage
`data_string`	ptr to character string	String containing characters copied beginning at the presentation space position identified by the input parameter `return_code` in the connected presentation space. The number of copied bytes is indicated in the input parameter `length`.
`return_code`	ptr to integer	Status of function call. Supported `return_code` values are listed in Table 4-19.

Table 4-19 Copy Presentation Space to String return_code Values

Value	Name	Explanation
0	HRC_SUCCESSFUL	Successful copy; presentation space is unlocked and ready for input.
1	HRC_PS_ID_INVALID	Unsuccessful copy; specified presentation space does not exist.
2	HRC_PARAMETER_ERROR	Unsuccessful copy; error in specifying parameters.
4	HRC_PS_BUSY	Successful copy; presentation space is busy.
5	HRC_FUNCTION_INHIBITED	Successful copy; presentation space is locked.
7	HRC_PS_POSITION_INVALID	Unsuccessful copy; the presentation space position is not valid.
9	HRC_SYSTEM_ERROR	Unsuccessful copy; system error encountered.

ATTR/NOATTR, EAB/NOEAB and XLATE/NOXLATE affect the operation of Copy Presentation to String.

Comments

Memory for the parameter data_string must be preallocated before using this function.

See Also

Connect Presentation Space, Copy Presentation Space, Set Session Parameters

4.7 Copy String to Field (33)

Synopsis

```
#include          “ehllapi.h”

function = HFUN_COPY_STRING_TO_FIELD;   /* function = 33          */
strcpy(data_string, “Hello World!”);    /* data_string to copy    */
length = 12;                           /* copy 6 bytes to string  */
return_code = 1;                       /* PS position of field    */
hllc (&function, data_string, &length, &return_code);
```

Description

Copies characters from a string provided by an EHLLAPI application into a field in the current connected presentation space. The number of characters to copy is specified in the `length` parameter. The field is identified by the presentation space position of the field, or the presentation space position of any character in the field, as specified in the `return_code` parameter. This function begins the copy at the first character in the field. The copy ends when the end of the field is reached, or when the target string is finished.

This function can copy characters from protected or unprotected fields, but it can only be used with field-formatted presentation spaces. Characters in the string are converted into EBCDIC characters by this function.

Prior Calls

Function Number (1) `Connect Presentation Space`.

Input Parameters

Table 4-20 Copy String to Field Inputs

Input	Input Type	Usage
function	ptr to integer	Function number (33).
data_string	ptr to character string	String containing data to the copy to target field.
length	ptr to integer	Maximum number of bytes to copy.
return_code	ptr to integer	Presentation space position of any byte in the target field. The copy starts at the first character of the field.

Output Parameters

Table 4-21 Copy String to Field Output

Output	Output Type	Usage
return_code	ptr to integer	Status of function call. Supported return_code values are listed in Table 4-22.

Table 4-22 Copy String to Field return_code Values

Value	Name	Explanation
0	HRC_SUCCESSFUL	Successful copy; presentation space is unlocked and ready for input.
1	HRC_PS_ID_INVALID	Unsuccessful copy; specified presentation space does not exist.
2	HRC_PARAMETER_ERROR	Unsuccessful copy; error in specifying parameters.
5	HRC_FUNCTION_INHIBITED	Successful copy; presentation space is locked.
6	HRC_DATA_ERROR	Warning: The input parameter length and the field length do not match. The data may be truncated.

Table 4-22 Copy String to Field return_code Values *(Continued)*

Value	Name	Explanation
7	HRC_PS_POSITION_INVALID	Unsuccessful copy; the presentation space position is not valid.
9	HRC_SYSTEM_ERROR	Unsuccessful copy; system error encountered.
24	HRC_PS_UNFORMATTED	Unsuccessful copy; presentation space is not field-formatted.

Session Parameters

STRLEN/STREOT and EOT=c affect the operation of Copy String to Field.

Comments

Keyboard mnemonics cannot be sent using the Copy String to Field function.

See Also

Connect Presentation Space, Set Session Parameters

4.8 Copy String to Presentation Space (15)

Synopsis

```
#include          “ehllapi.h”

function = HFUN_COPY_STRING_TO_PS;      /* function = 15          */
strcpy(data_string, “Hello World!”);    /* data_string allocated
*/
length = 12;                           /* copy 12 bytes to PS     */
return_code = 1;                       /* PS position to start at */
hllc (&function, data_string, &length, &return_code);
```

Description

Copies characters from a string provided by the EHLLAPI application into the current connected presentation space. The number of characters to copy is specified in the `length` parameter. The copy begins at the presentation position specified in the `return_code` parameter.

Characters in the string are converted into EBCDIC when copied into the presentation space.

Prior Calls

Function Number (1) `Connect Presentation Space`

Input Parameters

Table 4-23 `Copy String to Presentation Space` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number 15 (`Copy String to Presentation Space`).
`data_string`	ptr to character string	String containing data to copy into presentation space.
`length`	ptr to integer	Number of bytes to copy.
`return_code`	ptr to integer	Presentation space position at which to begin copy.

Output Parameters

Table 4-24 `Copy String to Presentation Space` Output

Output	Output Type	Usage
`return_code`	ptr to integer	Status of function call. Supported `return_code` values are listed in Table 4-25.

Table 4-25 Copy String to Presentation Space return_code Values

Value	Name	Explanation
0	HRC_SUCCESSFUL	Successful copy; presentation space is unlocked and ready for input.
1	HRC_PS_ID_INVALID	Unsuccessful copy; specified presentation space does not exist.
2	HRC_PARAMETER_ERROR	Unsuccessful copy; error in specifying parameters.
5	HRC_FUNCTION_INHIBITED	Successful copy; presentation space is locked.
6	HRC_DATA_ERROR	Warning: The input parameter length and the field length do not match. The data may be truncated.
7	HRC_PS_POSITION_INVALID	Unsuccessful copy; the presentation space position is not valid.
9	HRC_SYSTEM_ERROR	Unsuccessful copy; system error encountered.

Session Parameters

STRLEN/STREOT and EOT=c affect the operation of Copy String to Presentation Space.

Comments

Keyboard mnemonics cannot be sent using the Copy String to Field function.

See Also

Connect Presentation Space, Set Session Parameters

4.9 Disconnect Presentation Space (2)

Synopsis

```
#include          "ehllapi.h"
function = HFUN_DISCONNECT_PS;        /* function = 2          */
hllc (&function, data_string, &length, &return_code);
```

Description

Removes the connection between the EHLLAPI application and the current connected 3270 presentation space.

Prior Calls

Function Number (1) Connect Presentation Space

Input Parameters

Table 4-26 Disconnect Presentation Space Input

Input	Input Type	Usage
function	ptr to integer	Function number (2).

Output Parameters

Table 4-27 Disconnect Presentation Space Output

Output	Output Type	Usage
return_code	ptr to integer	Values listed in Table 4-28.

Table 4-28 `Disconnect Presentation Space return_code` Values

Value	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful disconnection; presentation space is unlocked and ready for input.
1	`HRC_PS_ID_INVALID`	Unsuccessful disconnection; specified presentation space does not exist.
9	`HRC_SYSTEM_ERROR`	Unsuccessful disconnection; system error encountered.

Session Parameters

Not applicable

Comments

This function does not reset the session parameters. Functions that require a current connected presentation space are not valid after executing this function (until a subsequent `Connect Presentation Space` call is successful).

See Also

`Connect Presentation Space`

4.10 Find Field Length (32)

Synopsis

```
#include          “ehllapi.h”

function = HFUN_FIND_FIELD_LENGTH;      /* function = 32          */
strcpy(data_string, “  “);             /* field len for this field */
return_code = 1;                        /* PS position of field    */
hllc (&function, data_string, &length, &return_code);
```

Description

Returns the length of a target field in the current connected presentation space. This function can be used for protected and unprotected fields, but it is only valid on field-formatted presentation spaces.

The length of the field is the number of characters from the beginning of the field to the next attribute byte.

Prior Calls

Function Number (1) `Connect Presentation Space`

Input Parameters

Table 4-29 `Find Field Length` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (32).
`data_string`	ptr to character string	Character string (two characters) indicating relation of the target field to the field indicated by the input parameter `return_code`. Possible values are listed in Table 4-30.
`length`	ptr to integer	Not applicable (2 is implicit).
`return_code`	ptr to integer	Presentation space position identifying a field in the current connected presentation space. May be any byte in the field used to anchor the `Find`.

Table 4-30 `Find Field Length data_string` Values

Value	Explanation
" "or "T "	This field
"P "	Previous field
"N "	Next field
"NP"	Next protected field

Table 4-30 `Find Field Length data_string` Values *(Continued)*

Value	Explanation
"NU"	Next unprotected field
"PP"	Previous protected field
"PU"	Previous unprotected field

Output Parameters

Table 4-31 `Find Field Length` Outputs

Output	Output Type	Usage
`length`	ptr to integer	Error when 0. Length of field when > 0.
`return_code`	ptr to integer	Status of function call. Supported `return_code` values are listed in Table 4-32.

Table 4-32 `Find Field Length return_code` Values

Value	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful.
1	`HRC_PS_ID_INVALID`	Unsuccessful; specified presentation space does not exist.
2	`HRC_PARAMETER_ERROR`	Unsuccessful; error in specifying parameters.
7	`HRC_PS_POSITION_INVALID`	Unsuccessful; the presentation space position is not valid.
9	`HRC_SYSTEM_ERROR`	Unsuccessful; system error encountered.
24	`HRC_STRING_NOT_FOUND`	Unsuccessful; field not found.
28	`HRC_FIELD_ZERO_LENGTH`	Unsuccessful; field is zero bytes in length.

Session Parameters

Not applicable

Comments

Except when " " or "T " is used as the data_string parameter, if the field found is the same as the field specified by the input parameter return_code, then the output parameter return_code indicates HRC_STRING_NOT_FOUND.

See Also

Connect Presentation Space, Set Session Parameters, Find Field Position

4.11 Find Field Position (31)

Synopsis

```
#include          "ehllapi.h"

function = HFUN_FIND_FIELD_POSITION;    /* function = 31          */
strcpy(data_string, "  ");            /* field pos for this field */
return_code = 1;                       /* PS position of field    */
hllc (&function, data_string, &length, &return_code);
```

Description

Returns the presentation space position of a target field in the current connected presentation space. This function can be used for protected and unprotected fields, but it is only valid on field-formatted presentation spaces.

Prior Calls

Function Number (1) Connect Presentation Space

Input Parameters

Table 4-33 `Find Field Position` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (31).
`data_string`	ptr to character string	Character string (2 characters) indicating relation of the target field to the field indicated by the input parameter `return_code`. Possible values are listed in Table 4-34.
`length`	ptr to integer	Not applicable (2 is implicit).
`return_code`	ptr to integer	Presentation space position identifying a field in the current connected presentation space. May be any byte in the field used to anchor the `Find`.

Table 4-34 `Find Field Position data_string` Values

Value	Explanation
“ “ or “T “	This field
“P “	Previous field
“N “	Next field
“NP”	Next protected field
“NU”	Next unprotected field
“PP”	Previous protected field
“PU”	Previous unprotected field

Output Parameters

Table 4-35 `Find Field Position` Outputs

Output	Output Type	Usage
`length`	ptr to integer	Error when 0. Position of field when > 0.
`return_code`	ptr to integer	Status of function call. Supported values are listed in Table 4-36.

Table 4-36 Find Field Position return_code Values

Value	Name	Explanation
0	HRC_SUCCESSFUL	Successful.
1	HRC_PS_ID_INVALID	Unsuccessful; specified presentation space does not exist.
2	HRC_PARAMETER_ERROR	Unsuccessful; error in specifying parameters.
7	HRC_PS_POSITION_INVALID	Unsuccessful; the presentation space position is not valid.
9	HRC_SYSTEM_ERROR	Unsuccessful; system error encountered.
24	HRC_STRING_NOT_FOUND	Unsuccessful; field not found.
28	HRC_FIELD_ZERO_LENGTH	Unsuccessful; field is zero bytes in length.

Session Parameters

Not applicable

Comments

Except when " " or "T " is used as the data_string parameter, if the field found is the same as the field specified by the input parameter return_code, then the output parameter return_code indicates HRC_STRING_NOT_FOUND.

See Also

Connect Presentation Space, Set Session Parameters, Find Field Length

4.12 Get Key (51)

Synopsis

```
#include          "ehllapi.h"

function = HFUN_GET_KEY;              /* function = 31          */
strcpy(data_string, "A      ");       /* ps is "A"              */
hllc (&function, data_string, &length, &return_code);
```

Description

Receives keystrokes from sessions with keystroke intercept enabled (see `Start Keystroke Intercept`). Received keystrokes can be processed, accepted, or rejected. Use this function to validate keystrokes into a presentation space.

Prior Calls

Function Number (50) `Start Keystroke Intercept`

Input Parameters

Table 4-37 `Get Key` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (51).
`data_string`	ptr to character string	Character string (8 characters) indicating the presentation space name in the first character.
`length`	ptr to integer	Not applicable (8 is implicit).

Table 4-38 `Get Key data_string` Values

Value	Explanation
Byte 0	Presentation space name, or a blank or null indicating the current connected presentation space
Bytes 1-7	Blanks (placeholders)

Note – When the presentation space is identified by a transient port, the presentation space name *must* be specified as a space.

Output Parameters

Table 4-39 `Get Key` Outputs

Output	Output Type	Usage
`data_string`	ptr to character string	String containing keystroke information.
`return_code`	ptr to integer	Status of function call.

Table 4-40 Get Key return_code Output Values

Value	Name	Explanation
0	HRC_SUCCESSFUL	Successful.
1	HRC_PS_ID_INVALID	Unsuccessful; specified presentation space does not exist.
5	HRC_FUNCTION_INHIBITED	Unsuccessful; only AID keys specified in Start Keystroke Intercept and non-AID keys are inhibited by this session type.
8	HRC_PROCEDURE_ERROR	Unsuccessful; Start Keystroke Intercept was not called previously.
9	HRC_SYSTEM_ERROR	Unsuccessful; system error encountered.
20	HRC_UNDEFINED_COMBINATION	Unsuccessful; undefined key combination typed.
25	HRC_KEYSTROKES_UNAVAILABLE	Unsuccessful; keystrokes not available on input queue.
31	HRC_KEYSTROKES_LOST	Unsuccessful; keystrokes lost due to input queue overflow.

Session Parameters

WAIT affects Get Key

Comments

A return_code value of HRC_KEYSTROKES_LOST indicates that the input queue size should be increased or Get Key should be invoked more frequently.

Supported Get Key return_code output values are listed in Table 4-40.

See Also

Send Key, Start Keystroke Intercept, Stop Keystroke Intercept, Set Session Parameters

4.13 Pause (18)

Synopsis

```
#include          "ehllapi.h"

function = HFUN_PAUSE;                 /* function = 31          */
length = 10;                           /* pause for 5 seconds    */
hllc (&function, data_string, &length, &return_code);
```

Description

Waits for a specified time period (indicated in half-seconds). This function can be ended before the time period has elapsed if a prior `Start Host Notification` has been called and the `IPAUSE` option has been selected using `Set Session Parameters`. In this case, events that occur in the current connected presentation space that correspond to those selected in `Start Host Notification` will end the `Pause`.

Prior Calls

Function Number (1) `Connect Presentation Space`

Input Parameters

Table 4-41 `Pause` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (18).
`length`	ptr to integer	Length of pause duration in half seconds.

Output Parameters

Table 4-42 `Pause` Output

Output	Output Type	Usage
`return_code`	ptr to integer	Status of function call.

Table 4-43 Pause return_code Values

Value	Name	Explanation
0	HRC_SUCCESSFUL	Successful.
9	HRC_SYSTEM_ERROR	Unsuccessful; system error encountered.
26	HRC_PS_UPDATED	Warning: Presentation space updated or OIA updated before pause elapsed.

Session Parameters

FPAUSE/IPAUSE affects Pause.

Comments

If the IPAUSE option is selected and the Pause has been satisfied by a host event, use Query Host Update to identify the host event and clear the notification queue before the next Pause.

See Also

Start Host Notification, Stop Host Notification, Query Host Update, Set Session Parameters.

4.14 Post Intercept Status (52)

Synopsis

```
#include          "ehllapi.h"

function = HFUN_POST_INTERCEPT_STATUS;   /* function = 52          */
strcpy(data_string, "AR");              /* ps name and reject key  */
hllc (&function, data_string, &length, &return_code);
```

Description

Accepts or rejects a keystroke retrieved using Get Key. If the keystroke is rejected, the sun3270 presentation space issues a beep.

Prior Calls

Function Number (50) `Start Keystroke Intercept`

Input Parameters

Table 4-44 `Post Intercept Status` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (23).
`data_string`	ptr to character string	String identifying the presentation space in the first character and whether the keystroke is accepted or rejected in the second character. Supported `Post Intercept Status data_string` values are listed in Table 4-45.
`length`	ptr to integer	Not applicable (2 is implicit).

Table 4-45 `Post Intercept Status data_string` Values

Value	Explanation
Byte 0	Presentation space name, or a blank or null to indicate the current connected presentation space.
Byte 1	'A' specifies accept key. 'R' specifies reject key.

Note – When the presentation space is identified by a transient port, the presentation space name *must* be specified as a space.

Output Parameters

Table 4-46 `Post Intercept Status` Output

Output	Output Type	Usage
`return_code`	ptr to integer	Status of function call. Supported `return_code` values are listed in Table 4-47.

Table 4-47 Post Intercept Status return_code Values

Value	Name	Explanation
0	HRC_SUCCESSFUL	Successful.
1	HRC_PS_ID_INVALID	Unsuccessful; specified presentation space does not exist
2	HRC_PARAMETER_ERROR	Unsuccessful; error in specifying parameters.
8	HRC_PROCEDURE_ERROR	Unsuccessful; Start Keystroke Intercept was not called previously.
9	HRC_SYSTEM_ERROR	Unsuccessful; system error encountered.

Session Parameters

Not applicable

Comments

Use Get Key to retrieve keystrokes. Use Post Intercept Status to accept or reject keystrokes. Use Send Key to replace intercepted keystrokes.

See Also

Get Key, Send Key, Start Intercept Status, Stop Keystroke Intercept

4.15 Query Cursor Location (7)

Synopsis

```
#include          "ehllapi.h"

function = HFUN_QUERY_CURSOR_LOCATION;   /* function = 7          */
hllc (&function, data_string, &length, &return_code);
```

Description

Returns the position of the cursor in the current connected presentation space.

Prior Calls

Function Number (1) `Connect Presentation Space`

Input Parameters

Table 4-48 `Query Cursor Location` Input

Input	Input Type	Usage
`function`	ptr to integer	Function number

Output Parameters

Table 4-49 `Query Cursor Location` Outputs

Output	Output Type	Usage
`length`	ptr to integer	Presentation space position of cursor.
`return_code`	ptr to integer	Status of function call. Supported values are listed in Table 4-50.

Table 4-50 `Query Cursor Location return_code` Values

Value	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful.
1	`HRC_PS_ID_INVALID`	Unsuccessful; specified presentation space does not exist.
9	`HRC_SYSTEM_ERROR`	Unsuccessful; system error encountered.

Session Parameters

Not applicable

Comments

None

See Also

`Set Cursor`

4.16 Query Field Attribute (14)

Synopsis

```
#include          "ehllapi.h"
function = HFUN_QUERY_FIELD_ATTRIBUTE;   /* function = 14          */
length = 1;                              /* PS position of field   */
hllc (&function, data_string, &length, &return_code);
```

Description

Returns the attribute byte of the field containing the specified presentation position.

Prior Calls

Function Number (1) Connect Presentation Space

Input Parameters

Table 4-51 Query Field Attribute Inputs

Input	Input Value	Usage
function	ptr to integer	Function number (14).
length	ptr to integer	Presentation space position of field. May be the presentation space of any byte in the target field.

Output Parameters

Table 4-52 Query Field Attribute Outputs

Output	Output Type	Usage
\length	ptr to integer	Attribute byte value, or 0 if error.
return_code	ptr to integer	Status of function call. Supported values are listed in Table 4-53.

Table 4-53 Query Field Attribute return_code Values

Value	Name	Explanation
0	HRC_SUCCESSFUL	Successful.
1	HRC_PS_ID_INVALID	Unsuccessful; specified presentation space does not exist.
7	HRC_PS_POSITION_INVALID	Unsuccessful; the presentation space position is not valid.
9	HRC_SYSTEM_ERROR	Unsuccessful; system error encountered.
24	HRC_PS_UNFORMATTED	Unsuccessful; presentation space is unformatted.

Session Parameters

Not applicable

Comments

None

See Also

Find Field

4.17 Query Host Update (24)

Synopsis

```
#include          "ehllapi.h"
function = HFUN_QUERY_HOST_UPDATE;      /* function = 24          */
strcpy(data_string, "A");              /* presentation space name */
hllc (&function, data_string, &length, &return_code);
```

Description

Determines if the presentation space or the OIA has been updated since the previous Start Host Notification or Query Host Update.

Prior Calls

Function Number (23) `Start Host Notification`

Input Parameters

Table 4-54 `Query Host Update` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (7).
`data_string`	ptr to character string	String identifying the presentation space to check for host updates.
`length`	ptr to integer	Not applicable (1 is implicit).

Note – When the presentation space is identified by a transient port, the presentation space name *must* be specified as a space.

Output Parameters

Table 4-55 `Query Host Update` Output

Output	Output Type	Usage
`return_code`	ptr to integer	Status of function call. Supported values of `return_code` are listed in Table 4-56.

Table 4-56 `Query Host Update return_code` Values

Value	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful.
1	`HRC_PS_ID_INVALID`	Unsuccessful; specified presentation space does not exist.
9	`HRC_SYSTEM_ERROR`	Unsuccessful; system error encountered.
21	`HRC_OIA_UPDATED`	Successful; OIA updated.
22	`HRC_PS_ONLY_UPDATED`	Successful; presentation space updated.
23	`HRC_PS_OIA_UPDATED`	Successful; both presentation space and OIA updated.

Session Parameters

Not applicable

Comments

None

See Also

`Start Host Notification`, `Stop Host Notification`, `Pause`

4.18 Query Session Status (22)

Synopsis

```
#include          “ehllapi.h”

function = HFUN_QUERY_SESSION_STATUS;   /* function = 22          */
strcpy(data_string, “A                “;   /* PS name and room    */
length = 18;
hllc (&function, data_string, &length, &return_code);
```

Description

Obtains session information.

Prior Calls

None

Input Parameters

Table 4-57 `Query Session Status` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (22).
`data_string`	ptr to character string	String identifying the presentation space to query and room for return values; total 18 bytes.
`length`	ptr to integer	Length of data string (18 bytes).

Note – When the presentation space is identified by a transient port, the presentation space name *must* be specified as a space.

Output Parameters

Table 4-58 `Query Session Status` Outputs

Output	Output Type	Usage
`data_string`	ptr to character string	String identifying session-supported values are listed in Table 4-59.
`return_code`	ptr to integer	Status of function call. Supported values are listed in Table 4-60.

Table 4-59 `Query Session Status data_string` Values

Byte	Explanation
0	Presentation space name
1-8	Presentation space name (again)
9	Session type
10	Session characteristics
11	Number of rows in presentation space (binary number)
13	Number of columns in presentation space (binary number)
15-16	Presentation space code page (binary number)

Table 4-60 `Query Session Status return_code` Values

Value	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful.
1	`HRC_PS_ID_INVALID`	Unsuccessful; specified presentation space does not exist.
2	`HRC_PARAMETER_ERROR`	Unsuccessful; error in specifying parameters.
9	`HRC_SYSTEM_ERROR`	Unsuccessful; system error encountered.

Session Parameters

Not applicable

Comments

Memory for the parameter `data_string` must be preallocated before using this function.

4.19 Query Sessions (10)

Synopsis

```
#include          "ehllapi.h"

function = HFUN_QUERY_SESSIONS;         /* function = 10          */
                                      /* data_string allocated    */
length = 8*12;
hllc (&function, data_string, &length, &return_code);
```

Description

Obtain session information on EHLLAPI sessions with short names in the range A through H.

Prior Calls

None

Input Parameters

Table 4-61 `Query Session` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (22).
`data_string`	ptr to character string	Array of one 12-byte entry for each EHLLAPI 3270 session; total 8*12 bytes.
`length`	ptr to integer	8*12 bytes = 96.

Output Parameters

Table 4-62 `Query Session` Outputs

Output	Output Type	Usage
`data_string`	ptr to character string	Array of 12 byte entries, one entry for each active EHLLAPI 3270 session. Each entry has the form outlined in Table 4-63.
`function`	ptr to integer	Function number (22).
`length`	ptr to integer	Number of active EHLLAPI 3270 sessions.
`return_code`	ptr to integer	Status of function call. Supported values are listed in Table 4-64.

Table 4-63 `Query Session data_string` Format

Byte	Explanation
0	Presentation space name
1-8	Presentation space name (again)
9	Session type
10-11	3270 presentation space size (binary number)

Table 4-64 `Query Session return_code` Format

Value	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful.
2	`HRC_PARAMETER_ERROR`	Unsuccessful; error in specifying parameters.
9	`HRC_SYSTEM_ERROR`	Unsuccessful; system error encountered.

Session Parameters

Not applicable

Comments

Memory for the parameter `data_string` must be preallocated before using this function.

4.20 Query System (20)

Synopsis

```
#include          "ehllapi.h"

function = HFUN_QUERY_SESSION_STATUS;    /* function = 20          */
                                         /* data_string allocated  */
hllc (&function, data_string, &length, &return_code);
```

Description

Obtains system information

Prior Calls

None

Input Parameters

Table 4-65 `Query System` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (20).
`data_string`	ptr to character string	String allocated for receiving system information; total 35 bytes.
`length`	ptr to integer	Not applicable (35 bytes is implicit).

Output Parameters

Table 4-66 `Query System` Outputs

Output	Output Type	Usage
`data_string`	ptr to character string	35-byte string identifying session values, as listed in Table 4-67.
`return_code`	ptr to integer	Status of function call. Supported `return_code` values are listed in Table 4-68.

Table 4-67 `Query System Output data_string` Format

Byte	Explanation
0	EHLLAPI version number
1-2	EHLLAPI level number
3-8	EHLLAPI date (month, date, year)
9-11	Reserved
12	Hardware base (U=unable to determine)
13	Program type (S=Sun)
14-15	Reserved
16-17	Sun SNA version
18	Reserved
19-22	Error code 1: ASCII string representing hex error
23-26	Error code 2: ASCII string representing hex error
27-28	System model number
29-30	NLS type
31	Monitor
32-34	Reserved

Table 4-68 `Query System Output return_code` Format

Value	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful.
1	`HRC_PS_ID_INVALID`	Unsuccessful; specified presentation space does not exist.
2	`HRC_PARAMETER_ERROR`	Unsuccessful; error in specifying parameters.
9	`HRC_SYSTEM_ERROR`	Unsuccessful; system error encountered.

Session Parameters

Not applicable

Comments

Memory for the parameter `data_string` must be preallocated before using this function.

4.21 Receive File (91)

Synopsis

```
#include          “ehllapi.h”

function = HFUN_RECEIVE_FILE;          /* function = 91          */
strcpy(data_string, “myfile hostfile”);  /* Sun pcft parameters  */
length = 15;                          /* copy 15 bytes to string  */
hllc (&function, data_string, &length, &return_code);
```

Description

`Receive File` transfers a file from the IBM host to the local UNIX system using the 3270 session. This function invokes the Sun PC File Transfer program (`pcft receive`). For more detailed information about `pcft`, refer to the *SunLink Client 3270 9.0 Configuration and User's Guide.*

Prior Calls

None

Input Parameters

Table 4-69 `Receive File` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (91).
`data_string`	ptr to character string	File transfer parameters. Use the Sun PC File Transfer parameters. This function prefixes `pcft receive` to this string.
`length`	ptr to integer	Number of bytes in string.

Output Parameters

Table 4-70 `Receive File` Output

Output	Output Type	Usage
`return_code`	ptr to integer	Status of function call. Supported `return_code` values are listed in Table 4-71.

Table 4-71 `Receive File return_code` Values

Value	Name	Explanation
2	`HRC_PARAMETER_ERROR`	Unsuccessful; error in specifying parameters.
3	`HRC_FT_COMPLETE`	Successful; file transfer complete.
9	`HRC_SYSTEM_ERROR`	Unsuccessful; system error encountered.
27	`HRC_FT_TERMINATED`	Unsuccessful; file transfer terminated by a Control-C or timeout.

Session Parameters

`QUIET/NOQUIET`, `TIMEOUT=C`,`STRLEN/STREOT`, and `EOT=C` affect the operation of `Receive File`.

Comments

As with Sun's PC File Transfer program (`pcft`), your EHLLAPI program must bring the corresponding IBM host application to the appropriate state before initiating a file transfer. If you are logged in to the IBM host application TSO, for example, TSO should be displaying the `READY` string.

See Also

`Send File`, `Set Session Parameter`

4.22 Release (12)

Synopsis

```
#include          "ehllapi.h"

function = HFUN_RELEASE;              /* function = 12          */
hllc (&function, data_string, &length, &return_code);
```

Description

Unlocks keyboard of the currently connected presentation space reserved by a previous call to Reserve.

Prior Calls

Function Number (1) Connect Presentation Space

Input Parameters

Table 4-72 Release Input

Input	Input Type	Usage
function	ptr to integer	Function number (12).

Output Parameters

Table 4-73 Release Output

Output	Output Type	Usage
return_code	ptr to integer	Status of function call. Supported return_code values are listed in Table 4-74

Table 4-74 `Release return_code` Values

Value	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful.
1	`HRC_PS_ID_INVALID`	Unsuccessful; specified presentation space does not exist.
9	`HRC_SYSTEM_ERROR`	Unsuccessful; system error encountered.

Session Parameters

Not applicable

Comments

Execute `Release` after reserving a keyboard or the keyboard remains locked until a subsequent `Reset System` or `Disconnect Presentation Space` is issued.

See Also

`Reserve`, `Reset System`, `Disconnect Presentation Space`

4.23 Reserve (11)

Synopsis

```
#include        "ehllapi.h"

function = HFUN_RESERVE;              /* function = 11          */
hllc (&function, data_string, &length, &return_code);
```

Description

Locks keyboard of the currently connected presentation space by blocking input from the terminal operator. The keyboard remains locked until one of these functions is executed: `Release`, `Reset System`, `Disconnect Presentation Space`.

Prior Calls

Function Number (1) `Connect Presentation Space`

Input Parameters

Table 4-75 `Reserve` Input

Input	Input Type	Usage
`function`	ptr to integer	Function number (11).

Output Parameters

Table 4-76 `Reserve` Output

Output	Output Type	Usage
`return_code`	ptr to integer	Status of function call. Supported values of `return_code` are listed in Table 4-77.

Table 4-77 `Reserve return_code` Values

Value	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful.
1	`HRC_PS_ID_INVALID`	Unsuccessful; specified presentation space does not exist.
9	`HRC_SYSTEM_ERROR`	Unsuccessful; system error encountered.

Session Parameters

Not applicable

Comments

Use this function to prevent a terminal operator from interfering with the EHLLAPI application processing.

See Also

`Release`, `Reset System`, `Disconnect Presentation Space`

4.24 Reset System (21)

Synopsis

```
#include          "ehllapi.h"
function = HFUN_RESET_SYSTEM;          /* function = 21          */
hllc (&function, data_string, &length, &return_code);
```

Description

Reinitializes the EHLLAPI program to its initial state. Session parameters are set to their default values, event notification is stopped, keyboards are released, keystroke intercept is disabled, and current presentation space is disconnected.

Prior Calls

None

Input Parameters

Table 4-78 `Reset System` Input

Input	Input Type	Usage
`function`	ptr to integer	Function number (21).

Output Parameters

Table 4-79 `Reset System` Output

Output	Output Type	Usage
`return_code`	ptr to integer	Status of function call. Supported values are listed in Table 4-80.

Table 4-80 `Reset System return_code` Values

Value	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful.
1	`HRC_PS_ID_INVALID`	Unsuccessful; specified presentation space does not exist.
9	`HRC_SYSTEM_ERROR`	Unsuccessful; system error encountered.

Session Parameters

Not applicable

Comments

None

4.25 Search Field (30)

Synopsis

```
#include          “ehllapi.h”

function = HFUN_SEARCH_FIELD;          /* function = 30          */
strcpy(data_string, “Hello World!”);    /* find string in field
*/
length = 12;                           /* copy 6 bytes to string   */
return_code = 1;                       /* PS position of field     */
hllc (&function, data_string, &length, &return_code);
```

Description

Searches the identified field for the string specified by the EHLLAPI program. The number of characters to copy is specified in the `length` parameter. The field is identified by the presentation space position of the field, or the presentation space position of any character in the field, as specified in the `return_code` parameter. The presentation space position of the first character of the string in the target field is returned in the output parameter `length`.

This function can search for strings from protected or unprotected fields, but it can only be used with field-formatted presentation spaces.

Prior Calls

Function Number (1) `Connect Presentation Space`

Input Parameters

Table 4-81 `Search Field` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (30).
`data_string`	ptr to character string	Target data string to find.
`length`	ptr to integer	Number of bytes in string.
`return_code`	ptr to integer	Presentation space position of any byte in the target field. The find starts with the first character.

Output Parameters

Table 4-82 `Search Field` Outputs

Output	Output Type	Usage
`length`	ptr to integer	Presentation space position indicating location of string. 0 means error.
`return_code`	ptr to integer	Status of function call. Supported values are listed in Table 4-83.

Table 4-83 `Search Field return_code` Values

Value	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful; presentation space is unlocked and ready for input.
1	`HRC_PS_ID_INVALID`	Unsuccessful; specified presentation space does not exist.
2	`HRC_PARAMETER_ERROR`	Unsuccessful; error in specifying parameters.

Table 4-83 Search Field return_code Values *(Continued)*

Value	Name	Explanation
7	HRC_PS_POSITION_INVALID	Unsuccessful; the presentation space position is not valid.
9	HRC_SYSTEM_ERROR	Unsuccessful; system error encountered.
24	HRC_STRING_NOT_FOUND	Unsuccessful; presentation space is not field-formatted, or search string is not found.

Session Parameters

SRCHALL/SRCHFROM, STRLEN/STREOT, SRCHFRWD/SRCHBKWD, and EOT=c affect the operation of Search Field.

Comments

Use these session parameters to control the search conditions: SRCHALL/SRCHFROM and SRCHFRWD/SRCHBKWD.

See Also

Search Presentation Space, Set Session Parameters

4.26 Search Presentation Space (6)

Synopsis

```
#include          "ehllapi.h"

function = HFUN_SEARCH_PS;             /* function = 6           */
bcopy(data_string, "Hello World!");     /* find string in PS      */
length = 12;                           /* copy 6 bytes to string  */
return_code = 1;                       /* PS pos to start search  */
hllc (&function, data_string, &length, &return_code);
```

Description

Searches the currently connected presentation space for the string specified by the EHLLAPI program. The number of characters to copy is specified in the length parameter. The presentation space beginning position is specified in

the return_code parameter. The presentation space position of the first character of the string in the presentation space is returned in the output parameter length.

Prior Calls

Function Number (1) Connect Presentation Space

Input Parameters

Table 4-84 Search Presentation Space Inputs

Input	Input Type	Usage
function	ptr to integer	Function number (6).
data_string	ptr to character string	Target data string to find.
length	ptr to integer	Number of bytes in string.
return_code	ptr to integer	Presentation space position at which to begin search.

Output Parameters

Table 4-85 Search Presentation Space Outputs

Output	Output Type	Usage
length	ptr to integer	Presentation space position indicating location of string. 0 means error.
return_code	ptr to integer	Status of function call. Supported return_code values are listed in Table 4-86.

Table 4-86 Search Presentation Space return_code Values

Value	Name	Explanation
0	HRC_SUCCESSFUL	Successful; presentation space is unlocked and ready for input.
1	HRC_PS_ID_INVALID	Unsuccessful; specified presentation space does not exist.
2	HRC_PARAMETER_ERROR	Unsuccessful; error in specifying parameters.

Table 4-86 Search Presentation Space return_code Values *(Continued)*

Value	Name	Explanation
7	HRC_PS_POSITION_INVALID	Unsuccessful; the presentation space position is not valid.
9	HRC_SYSTEM_ERROR	Unsuccessful; system error encountered.
24	HRC_STRING_NOT_FOUND	Unsuccessful; presentation space is not field-formatted, or search string is not found.

Session Parameters

SRCHALL/SRCHFROM, STRLEN/STREOT, SRCHFRWD/SRCHBKWD, and EOT=c affect the operation of Search Presentation Space.

Comments

Use these session parameters to control the search conditions: SRCHALL/SRCHFROM and SRCHFRWD/SRCHBKWD.

See Also

Set Session Parameters

4.27 Send File (90)

Synopsis

```
#include          "ehllapi.h"

function = HFUN_SEND_FILE;             /* function = 90          */
strcpy(data_string, "myfile hostfile"); /* Sun pcft parameters */
length = 15;                           /* copy 15 bytes to string */
hllc (&function, data_string, &length, &return_code);
```

Description

Send File transfers a file to the IBM host from the local UNIX system using the 3270 session. This function invokes the Sun PC File Transfer program (pcft send). For more detailed information about pcft, refer to the *SunLink Client 3270 9.0 Configuration and User's Guide*.

Prior Calls

None

Input Parameters

Table 4-87 Send File Inputs

Input	Input Type	Usage
function	ptr to integer	Function number (90).
data_string	ptr to character string	File transfer parameters. Use the Sun PC File Transfer parameters. This function prefixes "pcft send" to this string.
length	ptr to integer	Number of bytes in a string.

Output Parameters

Table 4-88 Send File Output

Output	Output Type	Usage
return_code	ptr to integer	Status of function call. Supported return_code values are listed in Table 4-89

Table 4-89 Send File return_code Values

Value	Name	Explanation
2	HRC_PARAMETER_ERROR	Unsuccessful; error in specifying parameters.
3	HRC_FT_COMPLETE	Successful; file transfer complete.
9	HRC_SYSTEM_ERROR	Unsuccessful; system error encountered.
27	HRC_FT_TERMINATED	Unsuccessful; file transfer terminated by a Control-C or timeout.

Session Parameters

QUIET/NOQUIET, TIMEOUT=c, STRLEN/STREOT, and EOT=c affect the operation of Send File.

Comments

Note – As with Sun PC File Transfer program (`pcft`), your EHLLAPI program must bring the corresponding IBM host application to the appropriate state before initiating a file transfer. If you are logged in to the IBM host application TSO, for example, TSO should be displaying the `READY` string.

See Also

`Receive File`, `Set Session Parameters`

4.28 Send Key (3)

Synopsis

```
#include          “ehllapi.h”

function = HFUN_SEND_KEY;              /* function = 3           */
bcopy(data_string, “Hello Worldly”);   /* keys to send (@E=enter) */
length = 14;                          /* length of data_string   */
hllc (&function, data_string, &length, &return_code);
```

Description

Sends a string of keystrokes (maximum 255 characters) to the currently connected presentation space. Use this function to emulate operator input.

Prior Calls

Function Number (1) `Connect Presentation Space`

Input Parameters

Table 4-90 `Send Key` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (`Send Key` is 3).
`data_string`	ptr to character string	Keys (and key mnemonics) to send.
`length`	ptr to integer	Number of bytes in a string.

Output Parameters

Table 4-91 Send Key Output

Output	Output Type	Usage
return_code	ptr to integer	Status of function call. Supported values of return_code are listed in Table 4-92.

Table 4-92 Send Key return_code Values

Value	Name	Explanation
0	HRC_SUCCESSFUL	Successful; presentation space is unlocked and ready for input.
1	HRC_PS_ID_INVALID	Unsuccessful; specified presentation space does not exist.
2	HRC_PARAMETER_ERROR	Unsuccessful; error in specifying parameters.
4	HRC_PS_BUSY	Unsuccessful; presentation space is busy.
5	HRC_FUNCTION_INHIBITED	Unsuccessful; presentation space is locked.
9	HRC_SYSTEM_ERROR	Unsuccessful; system error encountered.

Session Parameters

AUTORESET/NORESET, STRLEN/STREOT, EOT=c, and ESC=c affect the operation of Send Key.

Comments

Keystrokes cannot be sent to a busy or locked presentation space. Use the Wait function to check the presentation space status.

See Also

Get Key, Wait, Set Session Parameters

4.29 Set Cursor (40)

Synopsis

```
#include          "ehllapi.h"

function = HFUN_SET_CURSOR;            /* function = 40          */
return_code = 1;                      /* move cursor to 1st PS    */
hllc (&function, data_string, &length, &return_code);
```

Description

Moves the cursor to the specified position in the currently connected presentation space.

Prior Calls

Function Number (1) Connect Presentation Space

Input Parameters

Table 4-93 Set Cursor Input

Input	Input Type	Usage
function	ptr to integer	Function number (40).
return_code	ptr to integer	Position in presentation space at which the cursor is moved.

Output Parameters

Table 4-94 Set Cursor Output

Output	Output Type	Usage
return_code	ptr to integer	Status of function call. Supported values of return_code are listed in Table 4-95.

Table 4-95 Set Cursor return_code Values

Value	Name	Explanation
0	HRC_SUCCESSFUL	Successful; presentation space is unlocked and ready for input.
1	HRC_PS_ID_INVALID	Unsuccessful; specified presentation space does not exist.
2	HRC_PARAMETER_ERROR	Unsuccessful; error in specifying parameters.
7	HRC_PS_POSITION_INVALID	Unsuccessful; the presentation space position is not valid.
9	HRC_SYSTEM_ERROR	Unsuccessful; system error encountered.

Session Parameters

Not applicable

Comments

None

See Also

Query Cursor Location

4.30 Set Session Parameters (9)

Synopsis

```
#include          “ehllapi.h”
function = HFUN_SET_SESSION_PARAMETERS;  /* function = 9          */
strcpy(data_string, “STREOT EOT=!”);    /* parameters to set      */
length = 12;                           /* length of data string  */
hllc (&function, data_string, &length, &return_code);
```

Description

Changes specified session options in EHLLAPI. The option updates remain in effect until a subsequent `Set Session Parameters` changes the previously specified options or a `Reset System` call is made.

Prior Calls

None

Input Parameter

Table 4-96 `Set Session Parameter` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (9).
`data_string`	ptr to character string	List of options (space separated). See Table 4-99.
`length`	ptr to integer	Length of data string.

Output Parameters

Table 4-97 `Set Session Parameter` Output

Output	Output Type	Usage
`return_code`	ptr to integer	Status of function call; supported values of `return_code` are listed in Table 4-98.

Table 4-98 `Set Session Parameter return_code` Values

Value	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful; presentation space is unlocked and ready for input.
2	`HRC_PARAMETER_ERROR`	Unsuccessful; error in specifying parameters.
9	`HRC_SYSTEM_ERROR`	Unsuccessful; system error encountered.

Session Parameters

Not applicable

Comments

Table 4-99 lists the various session options and their meanings (the underlined options are the default values).

Table 4-99 `Set Session Parameter data_string` Values

Option	Explanation			
STRLEN	Length is passed for all strings.			
STREOT	Input data strings are terminated by an EOT character.			
EOT=c	Set the EOT character. Binary zero is the default. Do not leave blanks after the equals sign.			
SRCHALL	Search the entire presentation space or field.			
SRCHFROM	Search from specified presentation space (for SRCHFRWD) or search ends at a specified presentation space (for SRCHBKWD).			
SRCHFRWD	Search in ascending direction (relative to presentation space position).			
SRCHBKWD	Search in descending direction. Search is successful if the target string is found within the search bounds.			
NOATTRB	Convert all attribute bytes to blanks.			
ATTRB	Return attribute bytes.			
FPAUSE	Pause for full duration specified.			
IPAUSE	Interrupt pause if host event occurs.			
NOQUIET	Display SEND and RECEIVE file transfer messages.			
QUIET	Do not display SEND and RECEIVE file transfer messages.			
TIMEOUT=0	Break SEND or RECEIVE file transfers after 20 seconds.			
TIMEOUT=c	Break SEND or RECEIVE file transfers after specified time as noted below:			
	Character	*Value (mins)*	*Character*	*Value (mins)*
	1	0.5	8	4.0
	2	1.0	9	4.5
	3	1.5	J	5.0
	4	2.0	K	5.5
	5	2.5	L	6.0
	1	0.5	8	4.0
	2	1.0	9	4.5
ESC=c	Specifies the escape character for keystroke mnemonics. '@' is the default escape character. Do not leave a blank after the equals sign.			
AUTORESET	Prefix all strings of keys sent (`Send Key`) with a reset keystroke.			

Table 4-99 Set Session Parameter data_string Values *(Continued)*

Option	Explanation
NORESET	Do not prefix all strings of keys with a reset keystroke.
TROFF	Turn tracing off.
TRON	Turn tracing on. View hllc_trace file in the EHLLAPI in the current working directory of the application.
TWAIT	Wait: waits for XCLOCK or XSYSTEM status to clear. Waits up to 1 minute. Get Key: waits until key can be returned to the EHLLAPI application.
LWAIT	Wait: waits for XCLOCK or XSYSTEM status to clear; waits forever. Get Key: waits until key can be returned to the EHLLAPI application.
NWAIT	Wait: checks XCLOCK or XSYSTEM status and returns immediately. Get Key: returns immediately if no key available: return_code = HRC_KEYSTROKES_UNAVAILABLE.
NOEAB	Pass data only.
EAB	Pass extended attribute bytes. Make sure to allocate a data_string buffer twice as large as the presentation space.

See Also

Send Key, Wait, Copy Presentation Space, Search Presentation Space, Copy Presentation Space to String, Copy String to Presentation Space, Pause, Search Field, Copy String to Field, Copy Field to String, Get Key, Send File, Receive File

4.31 Start Host Notification (32)

Synopsis

```
#include          "ehllapi.h"

function = HFUN_START_HOST_NOTIFICATION;   /* function = 23       */
strcpy(data_string, "AB    ");         /* ps:ps and OIA updates   */
hllc (&function, data_string, &length, &return_code);
```

Description

Starts tracking the status changes to the presentation space and/or OIA of the specified presentation space. Use `Query Host Update` to determine the host event that occurred.

Prior Calls

None

Input Parameters

Table 4-100 `Start Host Notification` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (23).
`data_string`	ptr to character string	String identifying the presentation space in the first character and the type of events to inform the EHLLAPI application about in the second character. See Table 4-101.
`length`	ptr to integer	Not applicable (6 is implicit).

Table 4-101 `Start Host Notification data_string` Values

Value	Meaning
Byte 0	Presentation space name, or a blank or null to indicate the currently connected presentation space.
Byte 1	'B' specifies both presentation space and OIA updates. 'O' specifies OIA updates only. 'P' specifies presentation space updates only.
Byte 2-5	Reserved.

Note – When the presentation space is identified by a transient port, the presentation space name must be specified as a space (indicating the currently connected presentation space).

Output Parameters

Table 4-102 `Start Host Notification` Output

Output	Output Type	Usage
`return_code`	ptr to integer	Status of function call. Supported values are listed in Table 4-103.

Table 4-103 `Start Host Notification return_code` Values

Value	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful.
1	`HRC_PS_ID_INVALID`	Unsuccessful; specified presentation space does not exist.
2	`HRC_PARAMETER_ERROR`	Unsuccessful; error in specifying parameters.
9	`HRC_SYSTEM_ERROR`	Unsuccessful; system error encountered.

Session Parameters

Not applicable

Comments

Use `Pause` with the session option `IPAUSE` to determine when a host event has occurred. Use `Query Host Update` to determine what was updated.

See Also

`Query Host Update`, `Stop Host Notification`, `Pause`

4.32 Start Keystroke Intercept (50)

Synopsis

```
#include           “ehllapi.h”
function = HFUN_START_KEYSTROKE_INTERCEPT;   /* function = 50     */
strcpy(data_string, “AL    “);          /* ps name and all keys    */
length = 256;                          /* keystroke buffer size   */
hllc (&function, data_string, &length, &return_code);
```

Description

Captures keystrokes entered by the terminal operator into the specified presentation space. Keystrokes are saved until they are retrieved using `Get Key`. If the keystroke buffer overflows, subsequent keys are discarded. All keystrokes, or just `AID` keystrokes, can be intercepted.

Prior Calls

None

Input Parameters

Table 4-104 `Start Keystroke Intercept` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (23).
`data_string`	ptr to character string	String identifying the presentation space in the first character and the type of keystrokes to intercept in the second character. See Table 4-105.
`length`	ptr to integer	Not applicable (6 is implicit).

Table 4-105 `Start Keystroke Intercept data_string` Values

Values	Explanation
Byte 0	Presentation space name, or a blank or null to indicate the currently connected presentation space.
Byte 1	'D' specifies intercept `AID` keys only.
	'L' specifies intercept all keys.
Byte 2-5	Reserved

Note – When the presentation space is identified by a transient port, the presentation space name *must* be specified as a space (indicating the currently connected presentation space).

Output Parameters

Table 4-106 `Start Keystroke Intercept` Output

Output	Output Type	Usage
`return_code`	ptr to integer	Status of function call. Supported values of `return_code` are listed in Table 4-107.

Table 4-107 `Start Keystroke Intercept return_code` Values

Value	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful.
1	`HRC_PS_ID_INVALID`	Unsuccessful; specified presentation space does not exist.
2	`HRC_PARAMETER_ERROR`	Unsuccessful; error in specifying parameters.
4	`HRC_PS_BUSY`	Unsuccessful; presentation space is busy.
9	`HRC_SYSTEM_ERROR`	Unsuccessful; system error encountered.

Session Parameters

Not applicable

Comments

Use `Get Key` to retrieve keystrokes. Use `Post Intercept Status` to accept or reject keystrokes. Use `Send Key` to replace intercepted keystrokes.

See Also

`Get Key`, `Send Key`, `Post Intercept Status`,
`Stop Keystroke Intercept`

4.33 Stop Host Notification (25)

Synopsis

```
#include          “ehllapi.h”

function = HFUN_STOP_HOST_NOTIFICATION;    /* function = 25       */
strcpy(data_string, “A”);              /* ps name                 */
hllc (&function, data_string, &length, &return_code);
```

Description

Disables tracking of the status changes to the presentation space and/or OIA of the specified presentation space.

Prior Calls

Function Number (23) `Start Host Notification`

Input Parameters

Table 4-108 `Stop Host Notification` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (25).
`data_string`	ptr to character string	String identifying the presentation space.
`length`	ptr to integer	Not applicable (1 is implicit).

Note – When the presentation space is identified by a transient port, the presentation space name must be specified as a space (indicating the currently connected presentation space).

Output Parameters

Table 4-109 `Stop Host Notification` Output

Output	Output Type	Usage
`return_code`	ptr to integer	Status of function call. Supported values are listed in Table 4-110.

Table 4-110 `Stop Host Notification return_code` Values

Value	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful.
1	`HRC_PS_ID_INVALID`	Unsuccessful; specified presentation space does not exist.
8	`HRC_PROCEDURE_ERROR`	Unsuccessful; `Start Host Notification` was not called previously.
9	`HRC_SYSTEM_ERROR`	Unsuccessful; system error encountered.

Session Parameters

Not applicable

Comments

None

See Also

`Query Host Update`, `Start Host Notification`, `Pause`

4.34 Stop Keystroke Intercept (53)

```
#include          “ehllapi.h”

function = HFUN_STOP_KEYSTROKE_INTERCEPT;   /* function = 53      */
strcpy(data_string, “A”);                   /* ps name            */
hllc (&function, data_string, &length, &return_code);
```

Description

Disables the capture of keystrokes from the specified presentation space.

Prior Calls

Function Number (50) `Start Keystroke Intercept`

Input Parameters

Table 4-111 `Stop Keystroke Intercept` Inputs

Input	Input Type	Usage
`function`	ptr to integer	Function number (50).
`data_string`	ptr to character string	String identifying the presentation space.
`length`	ptr to integer	Not applicable (1 is implicit).

Note – When the presentation space is identified by a transient port, the presentation space name must be specified as a space (indicating the currently connected presentation space).

Output Parameters

Table 4-112 `Stop Keystroke Intercept` Output

Output	Output Type	Usage
`return_code`	ptr to integer	Status of function call. Supported values are listed in Table 4-113.

Table 4-113 Stop Keystroke Intercept return_code Values

Value	Name	Explanation
0	HRC_SUCCESSFUL	Successful.
1	HRC_PS_ID_INVALID	Unsuccessful; specified presentation space does not exist.
8	HRC_PROCEDURE_ERROR	Unsuccessful; Start Keystroke Intercept was not called previously.
9	HRC_SYSTEM_ERROR	Unsuccessful; system error encountered.

Session Parameters

Not applicable

Comments

None

See Also

Query Host Update, Start Host Notification, Pause

4.35 Wait(4)

Synopsis

```
#include          "ehllapi.h"

function = HFUN_WAIT;                  /* function = 4          */
hllc (&function, data_string, &length, &return_code);
```

Description

Waits for the host response status (XCLOCK or XSYSTEM) to clear the currently connected presentation space. Set Session Parameter options TWAIT, LWAIT, and NWAIT control the duration of the wait.

Prior Calls

Function Number (1) `Connect Presentation Space`

Input Parameters

Table 4-114 `Wait` Input

Input	Input Type	Usage
`function`	ptr to integer	Function number

Output Parameters

Table 4-115 `Wait` Output

Output	Output Type	Usage
`return_code`	ptr to integer	Status of function call. Supported values are listed in Table 4-116

Table 4-116 `Wait return_code` Values

Value	Name	Explanation
0	`HRC_SUCCESSFUL`	Successful.
1	`HRC_PS_ID_INVALID`	Unsuccessful; specified presentation space does not exist.
4	`HRC_PS_BUSY`	Presentation space is busy.
5	`HRC_FUNCTION_INHIBITED`	Presentation space is locked.
9	`HRC_SYSTEM_ERROR`	Unsuccessful; system error encountered.

Session Parameters

`TWAIT`/`LWAIT`/`NWAIT` affect the operation of `Wait`.

Comments

`Wait` returns `HRC_SUCCESSFUL` when the host unlocks the keyboard, but this return code does not necessarily indicate a transaction is complete. Use one of the search functions to verify expected keywords that are in the presentation space.

See Also

`Search Field`, `Search Presentation Space`, `Set Session Parameters`

Sample Programs A

This appendix contains three EHLLAPI programs. The first program, `query`, presents display buffer interrogation. The second two programs, `record` and `playback`, provide a basic session re-run capability.

A.1 `query.c`

Code Example A-1 Display Buffer Interrogation Sample *(1 of 6)*

```
/*
* query.c         Example EHLLAPI Program
*
* @(#)query.c   1.1 10/9/91
*
* This sample C program is designed to use EHLLAPI
* to obtain position and attribute information of
* the cursor location. The user is prompted to
* position the cursor accordingly, and is given
* the cursor position and a decoding of the field
* attribute byte at that position.
*
* This example program originated in the IBM
* document "IBM PC 3270 Emulation Program Entry Level
* Version 1.21 Programmer's Guide, High Level Language
* Application Program Interface."
*
*/
```

Code Example A-1 Display Buffer Interrogation Sample *(2 of 6)*

```
/* Include statement for standard input/output module */
#include <stdio.h>

/* HLLAPI Parameters */
int API_FUNC, API_LEN, API_RETC;
char API_STRING[255];


/*
 * This function returns the unsigned integer that
 * results from shifting byte right numbits. If a
 * negative number is passed through the numbits
 * parameter, byte is shifted left -numbits.
 *
 */

unsigned int shift (byte,numbits)
unsigned int byte;
int numbits;

  unsigned int result;

  if (numbits > 0)
    result = byte >> numbits;          /* right shift */
  else
    result = byte << -numbits;         /* left shift */

  return(result);



/*
 * This function is designed to extract the 8 'bits' that
 * define 'byte' and return them in the 'bits' array.
 *
 */

void byte_to_bits(byte, bits)
unsigned int byte;
short int bits[8];

  unsigned int andbyte;
  int i;
```

Code Example A-1 Display Buffer Interrogation Sample *(3 of 6)*

```
  unsigned int tempbyte;

  for (i = 7, andbyte = 1; i >= 0; --i, andbyte *= 2)
    /* zero out all but ith bit */
    tempbyte = byte & andbyte;
    /* shift ith bit to rightmost bit position */
    bits[i] = shift(tempbyte, 7-i);



/*
 * This function obtains information regarding
 * the attributes of the cursor location.
 *
 */

void get_info()

  /* cursor location info */
  int cursor_pos,
      cursor_row,
      cursor_col,
      attribyte;

  /* Connect to Host Session */
  API_FUNC = 1;

  strcpy(API_STRING, "A");
  API_LEN = 1;
  hllc(&API_FUNC, API_STRING, &API_LEN, &API_RETC);
  if (API_RETC != 0)
    printf("Warning: could not connect to Host.
    printf("Return Code: %d\n", API_RETC);
    return;


  /* Reserve Keyboard */
  API_FUNC = 11;
  hllc(&API_FUNC, API_STRING, &API_LEN, &API_RETC);

  /* Query Cursor Location */
  API_FUNC = 7;
  hllc(&API_FUNC, API_STRING, &API_LEN, &API_RETC);
```

Code Example A-1 Display Buffer Interrogation Sample *(4 of 6)*

```
if (API_RETC == 0)
  cursor_pos = API_LEN;
  printf("Cursor Location: Position %d\n", cursor_pos);
else
  printf("Warning: could not determine cursor location.\n");
  printf("Return Code: %d\n", API_RETC);
  return;


/* Convert Position to RowCol */
API_FUNC = 99;

strcpy(API_STRING, "AP");
API_RETC = cursor_pos;
hllc(&API_FUNC, API_STRING, &API_LEN, &API_RETC);
if (API_RETC == 0 || API_LEN == 0)
  Incorrect input error.\n"); return;
  return;
else if (API_RETC == 9998)
  printf("Warning: Invalid host id or Host never connected.
  return;
else if (API_RETC == 9999)
  printf("Warning: input error in Data String.\n");
  return;
else
  cursor_row = API_LEN;
  cursor_col = API_RETC;
  printf("Cursor Location: Row %d, Column %d.\n", cursor_row, cursor_col);
}

/* Query Field Attribute */
API_FUNC = 14;
API_RETC = cursor_pos;
hllc(&API_FUNC, API_STRING, &API_LEN, &API_RETC);
if (API_LEN == 0)
  printf("The screen is unformatted.\n");
  return;
else if (API_RETC != 0)
  printf("Could not determine the attribute byte.\n");
  return;
else
  int i;
  /* attribute byte */
  unsigned attribyte;
```

Code Example A-1 Display Buffer Interrogation Sample *(5 of 6)*

```
/* bit breakdown of attribute byte */
short int attribits[8];
short int attribit45;

/* attribute byte is returned in length parameter */
attribyte = API_LEN;

printf("Attribute byte (hex code): %x.\n", attribyte);
/* break down attribyte into attribits */
byte_to_bits(attribyte, attribits);

printf("Attribute byte (bin code): ");
for (i = 0; i < 7; ++i)
  printf("%d", attribits[i]);
printf("%d", attribits[i]); printf("\n");

/* Unprotected/Protected bit */
if (attribits[2] == 1)
  printf("Protected data field.\n");
else
  printf("Unprotected data field.\n");

/* Alpha/Numeric bit */
if (attribits[3] == 1)
  printf("Numeric data only.\n");
else
  printf("Alphanumeric data.\n");

/* Automatic skip */
if (attribits[2] == 1 && attribits[3] == 1)
  printf("Automatic skip.

/* Intensity/Selector Pen Detectability bit */
attribit45 = (2 * attribits[4]) + attribits[5];
if (attribit45 == 0)
  printf("Normal intensity.\n");
  printf("Not pen detectable.\n");
else if (attribit45 == 1)
  printf("Normal intensity.\n");
  printf("Pen detectable.\n");
else if (attribit45 == 2)
  printf("High intensity.\n");
  printf("Pen detectable.\n");
else if (attribit45 == 3)
```

Code Example A-1 Display Buffer Interrogation Sample *(6 of 6)*

```
      printf("Non-display.\n");
      printf("Not pen detectable.\n");



  /* Release Keyboard */
  API_FUNC = 12;
  hllc(&API_FUNC, API_STRING, &API_LEN, &API_RETC);

  /* Disconnect from Host Session */
  API_FUNC = 2;
  hllc(&API_FUNC, API_STRING, &API_LEN, &API_RETC);

/* end of function get_info */


main()

  char ch;

  /* Reset System */
  API_FUNC = 21;
  hllc(&API_FUNC, API_STRING, &API_LEN, &API_RETC);

  do
    printf("jump to your Host Session and move \n");
    printf("the cursor to the desired location.\n");
    printf("Press 'i<RETURN>' to obtain info, or \n");
    printf("      'q<RETURN>' to quit program.  --> ");
    ch = getchar();
    getchar();
    if (ch == 'i')
      get_info();
  while (ch != 'q');

  /* Reset System */
  API_FUNC = 21;
  hllc(&API_FUNC, API_STRING, &API_LEN, &API_RETC);
```

A.2 record.c

Code Example A-2 Keystroke Capture Program Sample *(1 of 12)*

```
/*
* record.c         Example EHLLAPI Program
*
* @(#)record.c   1.1 10/9/91
*
* The "record" example program captures keystrokes entered into a Sun
* 3270 Client presentation space, saves the keystrokes in a session file,
* and echoes the keystrokes back to the Sun 3720 Client presentation space.
*
*
* Usage: record <session_file>
*
*        where the default session_file is session.
*
*        To stop "record" enter ^C (Control-C).
*
*
* The "record" program's sibling, "playback", reads stored keystrokes from
* the session file and sends these keystrokes to a Sun 3270 Client presentation
* space.  For example, you can record your keystrokes as you login to and
* move through screens of an IBM mainframe application.  Later, you can
* start from the beginning, but this time invoke the "playback" program
* to login to and move through the same screens of an IBM mainframe
* application.
*
* These example programs illustrate several types of EHLLAPI functions:
*
*   record
*
*     set session parameters
*     connect presentation space
*     start keystroke intercept
*     get key
*     query cursor location
*     copy presentation space to string
*     send key
*     wait
*     query system
```

Code Example A-2 Keystroke Capture Program Sample *(2 of 12)*

```
*
*
*   playback
*
*     set session parameters
*     connect presentation space
*     disconnect presentation space
*     start host notification
*     query cursor location
*     search presentation space
*     pause
*     query host update
*     stop host notification
*     send key
*     wait
*     query system
*
* Moreover, these example programs illustrate the inherent difficulties
* in controlling a 3270 presentation space with EHLLAPI:
*
*      An EHLLAPI program interacts with a 3270 presentation space
*      in a fashion similar to a 3270 operator.  Like an operator,
*      an EHLLAPI program must constantly "peruse" the 3270
*      presentation space to recognize "familiar" screens, and then
*      enter the appropriate information.
*
* The "record" program intercepts keystrokes from a 3270 presentation space
* and attempts to remember the screen in which the keystrokes were
* entered.  The main processing loop is:
*
*  loop:
*      1.  get a keystroke typed into the 3270 PS
*      2.  determine the cursor location of the keystroke in the 3270 PS
*      3.  copy 80 bytes preceeding the cursor location into a string
*      4.  save the keystroke, cursor location, and the 80 bytes
*          of "context" in the session file.
*      5.  echo the saved keystroke to the 3270 PS
*   endloop.
*
* Later, the "playback" program will retrieve the keystrokes and the
* keystroke "context" from the session file.  The "playback" program
* will wait until it recognizes the keystroke "context" in the 3270 PS
```

Code Example A-2 Keystroke Capture Program Sample *(3 of 12)*

```
 * before sending the associated keystroke.
 *
 * The "record" and "playback" programs are not fool-proof!  In particular,
 * the "playback" program only uses 80 bytes to "recognize" a screen.
 *
 */

#include <stdio.h>
#include <signal.h>

/*
 * Include constants for EHLLAPI functions and
 * return codes.
 *
 */
#include "ehllapi.h"


#define CHECK_SIZE   80          /* number of bytes in keystroke "context" */

/*
 * General routine for hllc() interface.  Uses external variables
 * as input and output parameters.
 *
 * Remember: the hllc() parameters change function/meaning depending
 * on the "function" executed.  Please reference the EHLLAPI reference
 * manual for each function's usage.
 *
 */
static void call_hllapi();

static int function;             /* hllc(): parameter 1 */
static char data_string[128];    /* hllc(): parameter 2 */
static int data_length;          /* hllc(): parameter 3 */
static int return_code;          /* hllc(): parameter 4 */


/*
 * main
 *
 * Main loop processing for "record" program:
 *
```

Code Example A-2 Keystroke Capture Program Sample *(4 of 12)*

```
 *      1.  connect presentation space
 *      2.  start keystroke intercept
 *
 *   loop:
 *      3.  get keystroke
 *      4.  get keystroke context
 *      5.  save keystroke and keystroke context
 *      6.  send keystroke
 *   endloop.
 *
 * The main loop is only broken by a QUIT signal.
 *
 *
 */

main(argc, argv)
int argc;
char *argv[];

  static void quit_handler();   /* QUIT signal handling routine */
  char *playback_file;          /* playback file name */
  FILE *fp;                     /* playback file pointer reference */
  char keys[7];                 /* temporary variable for keystroke */
  int cursor_location;         /* temporary variable for keystroke location */
  char context_string[CHECK_SIZE+1];    /* temporary variable for "context" */

  /*
   * Catch quit signals
   *
   */
  signal(SIGQUIT, quit_handler);
  signal(SIGINT , quit_handler);

  /*
   * Set playback file name
   *
   */
  if (argc > 1)
    playback_file = argv[1];
  else
    playback_file = "session";
```

Code Example A-2 Keystroke Capture Program Sample *(5 of 12)*

```
 /*
 * Set HLLAPI tracing off
 *
 * You can turn tracing on with data_string = "TRON" and
 * data_length = 4.
 *
 */
function = HFUN_SET_SESSION_PARAMETERS;
strcpy(data_string, "TROFF");
data_length = 5;
call_hllapi();
if (return_code != HRC_SUCCESSFUL)
 printf("unable to set HLLAPI tracing on, rc = %d\n",
         return_code);
  exit(1);


/*
 * Connect to HLLAPI server
 *
 */
function = HFUN_CONNECT_PS;
strcpy(data_string, "A");
call_hllapi();
if (return_code != HRC_SUCCESSFUL)
  printf("unable to connect to 3270 Presentation Space, rc = %d\n",
         return_code);
  exit(1);


/*
 * Start Keystroke Intercept
 *
 */
function = HFUN_START_KEYSTROKE_INTERCEPT;
strcpy(data_string, "AL");
data_length = 256;              /* size of keystroke buffer */
call_hllapi();
if (return_code != HRC_SUCCESSFUL)
  printf("unable to intercept keystrokes, rc = %d\n",
         return_code);
```

Code Example A-2 Keystroke Capture Program Sample *(6 of 12)*

```
  exit(1);


/*
 * Open playback file
 *
 */
fp = fopen(playback_file, "w");
if (fp == NULL)
  printf("unable to open playback file %s\n", playback_file);
  exit(1);


/*
 * Begin keystroke processing loop!
 *
 */
while (1)
  /*
   * Get Keystroke
   *
   */
  function = HFUN_GET_KEY;
  strcpy(data_string, "A     ");
  data_length = 8;
  call_hllapi();
  if (return_code != HRC_SUCCESSFUL)
    printf("unable to get keystroke, rc = %d\n",
           return_code);
    exit(1);


  bcopy(&data_string[2], keys, 6);             /* save the keystroke */
  keys[6] = '';

  /*
   * Determine the keystroke "context"
   *
   */
  get_context(&cursor_location, CHECK_SIZE+1, context_string);

  /*
```

Code Example A-2 Keystroke Capture Program Sample *(7 of 12)*

```
     * Save the keystroke, cursor location, and keystroke context
     * in the session file.
     *
     */
    fprintf(fp, "%d\n", cursor_location);
    fprintf(fp, "%s\n", context_string);
    fprintf(fp, "%s\n", keys);

    /*
     * Send keystroke back to 3270 PS
     *
     */
    send_keys_wait(keys);




/*
 * get_context
 *
 * Determine what was "around" the entered keystroke so that
 * we can use this to determine when we should send the key
 * during playback.
 *
 */

get_context(cursor_ptr, str_len, ret_string)
int *cursor_ptr;
int str_len;
char *ret_string;

  /*
   * Get Cursor location
   *
   */
  function = HFUN_QUERY_CURSOR_LOCATION;
  call_hllapi();
  if (return_code != HRC_SUCCESSFUL)
    printf("unable to get cursor location, %d\n",
           return_code);
    exit(1);
```

Code Example A-2 Keystroke Capture Program Sample *(8 of 12)*

```
 /*
 * Set cursor position return parameter
 *
 */
*cursor_ptr = data_length;

/*
 * Copy str_len-1 characters which preceed the cursor.

* If we're near the beginning of the PS, copy
 * all the bytes from position 1 to the cursor location.
 *
 */
function = HFUN_COPY_PS_TO_STRING;

return_code = *cursor_ptr - (str_len - 1 -1);
if (return_code <= 0)
  return_code = 1;

data_length = *cursor_ptr - return_code + 1;

call_hllapi();

/*
 * Report error if copy unsuccessful: these return codes are
 * informational -- the copy was successful.
 *
 */
if (return_code != HRC_SUCCESSFUL
    && return_code != HRC_PS_BUSY
    && return_code != HRC_FUNCTION_INHIBITED)
  printf("unable to copy PS into string, rc =  %d\n",
         return_code);
  exit(1);


/*
 * Set "context" string return parameter
 *
 */
bcopy(data_string, ret_string, data_length);
ret_string[data_length] = '';
```

Code Example A-2 Keystroke Capture Program Sample *(9 of 12)*

```
 /*
 * send_keys_wait
 *
 * Send keys and wait for XCLOCK to clear.
 *
 */


send_keys_wait(keys)
char *keys;

  /*
   * Use STREOT and NORESET options for this session.
   * Use STREOT to demonstrate this useful option.  Use
   * NORESET because we are attempting to mimic the
   * operator's actions and he/she did not invisibly enter a
   * RESET key before the current keystroke.
   *
   */
  function = HFUN_SET_SESSION_PARAMETERS;
  strcpy(data_string, "STREOT NORESET");
  data_length = 14;
  call_hllapi();
  if (return_code != HRC_SUCCESSFUL)
    printf("unable to set session parameters, rc = %d\n",
           return_code);
    exit(1);


  /*
   * Send Key to the 3720 PS
   *
   */
  function = HFUN_SEND_KEY;
  strcpy(data_string, keys);
  call_hllapi();

  /*
   * Report error and exit if problem detected.
   *
   * Exception: we'll allow the user to get into local input error conditions
   * and get himself out.
```

Code Example A-2 Keystroke Capture Program Sample *(10 of 12)*

```
   *
   */
  if (return_code != HRC_SUCCESSFUL && return_code != HRC_FUNCTION_INHIBITED)
    printf("unable to send keys %s, rc = %d\n", data_string, return_code);
    exit(1);


  /*
   * Wait for the XCLOCK condition to clear.

   *
   */
  function = HFUN_WAIT;
  call_hllapi();

  if (return_code != HRC_SUCCESSFUL && return_code != HRC_FUNCTION_INHIBITED)
    printf("error waiting after sending keys, rc = %d\n", return_code);
    exit(1);


  /*
   * Reset STREOT to STRLEN
   *
   */
  function = HFUN_SET_SESSION_PARAMETERS;
  strcpy(data_string, "STRLEN");
  data_length = 6;
  call_hllapi();
  if (return_code != HRC_SUCCESSFUL)
    printf("unable to reset session parameters, rc = %d\n",
           return_code);
    exit(1);



/*
 * Call HLLAPI
 *
 * Invoke hllc() and handle system error conditions.
 *
 */
```

Code Example A-2 Keystroke Capture Program Sample *(11 of 12)*

```
static void
call_hllapi()


  /*
   * Call hllc() with external parameters.
   *
   */
  hllc(&function, data_string, &data_length, &return_code);


  /*
   * Handle system errors.
   *
   */
  if (return_code == HRC_SYSTEM_ERROR
      && function != HFUN_CONVERT_POSITION_OR_ROWCOL)

    printf("System error reported for function %d, function);

    function = HFUN_QUERY_SYSTEM;
    data_length = 35;
    hllc(&function, data_string, &data_length, &return_code);

    printf("Query system values:
    printf("EHLLAPI version number = %c\n",  data_string[0]);
    printf("EHLLAPI level number = %c%c\n", data_string[1], data_string[2]);
    printf("EHLLAPI date = %c%c/%c%c/%c%c\n",
                        data_string[3], data_string[4],
                        data_string[5], data_string[6],
                        data_string[7], data_string[8]);
    printf("3270 version = %c%c, data_string[16], data_string[17]);
    printf("hardware = %c, data_string[12]);
    printf("error code 1 = 0x%c%c%c%c\n",
          data_string[19], data_string[20], data_string[21], data_string[22]);
    printf("error code 2 = 0x%c%c%c%c\n",
          data_string[23], data_string[24], data_string[25], data_string[26]);
    exit(1);



/*
```

Code Example A-2 Keystroke Capture Program Sample *(12 of 12)*

```
 * quit_handler
 *
 * Just exit.  The 3720 EHLLAPI Server is pretty good-natured
 * and will cleanup for us (we could do a Reset System EHLLAPI function
 * call).
 *
 */

static void
quit_handler()


  exit(0);
```

A.3 playback.c

Code Example A-3 Keystroke Playback Program Sample *(1 of 13)*

```
/*
* playback.c         Example EHLLAPI Program
*
* @(#)playback.c
*
* The "playback" example program reads stored keystrokes from a session
* file and sends these keystrokes to a Sun 3270 Client presentation
* space.
*
* Usage: playback <session_file>
*
*        where the default session_file is session.
*
*
* The "playback" program's sibling, "record", captures keystrokes entered into
* a Sun 3270 Client presentation space, saves the keystrokes in a session file,
* and echoes the keystrokes back to the Sun 3270 Client presentation space.
* For example, you can record your keystrokes as you login to and
* move through screens of an IBM mainframe application.  Later, you can
* start from the beginning, but this time invoke the "playback" program
* to login to and move through the same screens of an IBM mainframe
* application.
*
* These example programs illustrate several types of EHLLAPI functions:
*
*   record
*
*     set session parameters
*     connect presentation space
*     start keystroke intercept
*     get key
*     query cursor location
*     copy presentation space to string
*     send key
*     wait
*     query system
*
*   playback
```

Code Example A-3 Keystroke Playback Program Sample *(2 of 13)*

```
*
*     set session parameters
*     connect presentation space
*     disconnect presentation space
*     start host notification
*     query cursor location
*     search presentation space
*     pause
*     query host update
*     stop host notification
*     send key
*     wait
*     query system
*
* Moreover, these example programs illustrate the inherent difficulties
* in controlling a 3270 presentation space with EHLLAPI:
*
*      An EHLLAPI program interacts with a 3270 presentation space
*      in a fashion similar to a 3270 operator.  Like an operator,
*      an EHLLAPI program must constantly "peruse" the 3270
*      presentation space to recognize "familiar" screens, and then
*      enter the appropriate information.
*
* The "playback" program attempts to sequentiallly match screens with
* keystrokes stored in the session file.  When "playback" recognizes
* a screen, or when "playback" times-out attempting to recognize a screen,
* the corresponding keystroke is sent to the 3270 presentation space.
* The main processing loop is:
*
*  loop:
*      1.  get a keystroke and keystroke context from the session file
*      2.  attempt to recognize the keystrokes screen from the context:
*
*          loop:
*              2b. get the current cursor location
*              2c. compare 80 bytes preceeding cursor location with
*                  the keystroke "context"
*              2d. if successful, then exit loop
*              2e. if unsuccessful, then pause and wait for a PS update
*              2f. if pause timed-out, then exit loop
*          endloop.
*
```

Code Example A-3 Keystroke Playback Program Sample *(3 of 13)*

```
 *      3.  send key to 3270 PS
 *  endloop.
 *
 *
 * The "record" and "playback" programs are not fool-proof!  In particular,
 * the "playback" program only uses 80 bytes to "recognize" a screen.
 *
 */

#include <stdio.h>

/*
 * Include constants for EHLLAPI functions and
 * return codes.
 *
 */
#include "ehllapi.h"


#define  PAUSE_TIME        15     /* secs to wait for a screen we recognize */


/*
 * General routine for hllc() interface.  Uses external variables
 * as input and output parameters.
 *
 * Remember: the hllc() parameters change function/meaning depending
 * on the "function" executed.  Please reference the EHLLAPI reference
 * manual for each function's usage.
 *
 */
static void call_hllapi();

static int function;
static char data_string[128];
static int data_length;
static int return_code;

/*
 * main
 *
 * Main loop processing for "playback" program:
```

Code Example A-3 Keystroke Playback Program Sample *(4 of 13)*

```
 *
 *      1.  connect presentation space
 *
 *   loop (while keystrokes in session file):
 *      2.  get keystroke context from session file
 *      3.  wait for screen which matches keystroke context (or time-out)
 *      4.  get keystroke from session file
 *      5.  send keystroke
 *   endloop.
 *
 *      6.  disconnect presentation space
 *
 *
 */

main(argc, argv)
int argc;
char *argv[];

  char key_string[10];
  char *playback_file;          /* playback file name */
  FILE *fp;                     /* playback file pointer reference */
  char cursor_string[10];       /* cursor location (unused by playback) */
  char context_string[128];     /* keystroke "context" string */
  int str_len;                  /* length of "context" string */

  /*
   * Set playback file name
   *
   */
  if (argc > 1)
    playback_file = argv[1];
  else
    playback_file = "session";

  /*
   * Set HLLAPI tracing off
   *
   * You can turn tracing on with data_string = "TRON" and
   * data_length = 4.
   *
   */
```

Code Example A-3 Keystroke Playback Program Sample *(5 of 13)*

```
function = HFUN_SET_SESSION_PARAMETERS;
 strcpy(data_string, "TROFF");
 data_length = 5;
 call_hllapi();
 if (return_code != HRC_SUCCESSFUL)
   printf("unable to set HLLAPI tracing on, rc = %d\n",

          return_code);
   exit(1);


 /*
  * Connect to HLLAPI server
  *
  */
 function = HFUN_CONNECT_PS;
 strcpy(data_string, "A");
 call_hllapi();
 if (return_code != HRC_SUCCESSFUL)
   printf("unable to connect to 3270 Presentation Space, rc = %d\n",
          return_code);
   exit(1);


 /*
  * Open playback file
  *
  */
 fp = fopen(playback_file, "r");
 if (fp == NULL)
   printf("unable to open playback file %s\n" , playback_file);
   exit(1);


 /*
  * Retrieve keys from file, wait for screen to match
  * expected "context", and send keys to 3270.
  *
  * Note: the cursor location (cursor_string) is ignored by
  * "playback".  To be extremely rigorous, we could make
  * sure that the current cursor location in the 3270 PS
  * matches the saved cursor location -- we don't.
```

Code Example A-3 Keystroke Playback Program Sample *(6 of 13)*

```
 *
 */
while (fgets(cursor_string, 10, fp))
  /*
   * Read context string from session file and attempt match.
   *
   */
  if (fgets(context_string, 128, fp))

    str_len = strlen(context_string);
    context_string[str_len-1] = '';  /* Strip off newline */

    /*
     * Attempt to match "context" string with the
     * appropriate 3270 PS.
     *
     */
    wait_for_context(str_len-1, context_string);


  /*
   * Read keystroke and send
   *
   */
  if (fgets(key_string, 10, fp))
    key_string[strlen(key_string)-1] = '';   /*Strip off newline */

    /*
     * Send keystroke to 3270 PS
     *
     */
    send_keys_wait(key_string);



/*
 * Disconnect from HLLAPI server
 *
 */
function = HFUN_DISCONNECT_PS;
strcpy(data_string, "A");
call_hllapi();
```

Code Example A-3 Keystroke Playback Program Sample *(7 of 13)*

```
  if (return_code != HRC_SUCCESSFUL)
    printf("unable to disconnect from 3270 Presentation Space, rc = %d\n",
           return_code);
    exit(1);


  /*
   * Close playback file
   *
   */
  fclose(fp);



/*
 * wait_for_context
 *
 * Wait until we find the context string matched,
 * or, until we timeout -- then assume there isn't
 * supposed to be a direct match and preceed at
 * our own risk.
 *
 */

wait_for_context(str_len, context_string)
int str_len;
char *context_string;

  int cursor_location;
  int original_pos;

  /*
   * Set Session Parameters for interruptible pause
   *
   */
  function = HFUN_SET_SESSION_PARAMETERS;
  strcpy(data_string, "IPAUSE SRCHFROM");
  data_length = 15;
  call_hllapi();
  if (return_code != HRC_SUCCESSFUL)
    printf("unable to set IPAUSE, rc = %d\n",
            return_code);
```

Code Example A-3 Keystroke Playback Program Sample *(8 of 13)*

```
  exit(1);


/*
 * Start Host Notification for PS or OIA updates
 *
 */
function = HFUN_START_HOST_NOTIFICATION;
strcpy(data_string, "AB");
  call_hllapi();
if (return_code != HRC_SUCCESSFUL)
  printf("unable to get start host notifications, rc = %d\n",
         return_code);
  exit(1);


/*
 * Constantly attempt to match the "context" string
 * with the current 3270 PS.
 *
 */
while (1)
  /*
   * Get Cursor location
   *
   */
  function = HFUN_QUERY_CURSOR_LOCATION;
  call_hllapi();
  if (return_code != HRC_SUCCESSFUL)
    printf("unable to get cursor location, rc = %d\n",
           return_code);
    exit(1);


  cursor_location = data_length;

  /*
   * Search str_len-1 characters which preceed the cursor.
   * If we're near the beginning of the PS, search
   * all the bytes from position 1 to the cursor location.
   *
   */
```

Code Example A-3 Keystroke Playback Program Sample *(9 of 13)*

```
    function = HFUN_SEARCH_PS;

    return_code = cursor_location - (str_len - 1);
    if (return_code <= 0)
      return_code = 1;

    original_pos = return_code;
    data_length = cursor_location - return_code + 1;

    strcpy(data_string, context_string);

    call_hllapi();

    /*
     * We found the expected string, so assume
     * context has been matched.
     *
     * Note: data_length returns the PS position of the match --
     * if it isn't the same as original expectations then we
     * found a match somewhere else in the PS (that's no good).
     *
     */
    if (return_code == HRC_SUCCESSFUL && data_length == original_pos)
      break;

    /*
     * Pause waiting for latest screen
     *
     */
    function = HFUN_PAUSE;
    data_length = PAUSE_TIME * 2;             /* in half-seconds */
    call_hllapi();

    /*
     * We timed-out, so assume context is what
     * it should be.
     *
     */
    if (return_code == HRC_SUCCESSFUL)
      break;

    /*
```

Code Example A-3 Keystroke Playback Program Sample *(10 of 13)*

```
     * Query Host Update to relieve notification queue
     *
     */
    function = HFUN_QUERY_HOST_UPDATE;
    strcpy(data_string, "A");
    call_hllapi();


  /*
   * Stop Host Notifications
   *
   */
  function = HFUN_STOP_HOST_NOTIFICATION;
  strcpy(data_string, "A");
  call_hllapi();
  if (return_code != HRC_SUCCESSFUL)
    printf("unable to get stop host notifications, rc = %d\n",
           return_code);
    exit(1);



/*
 * send_keys_wait
 *
 * Send keys and wait for XCLOCK to clear.
 *
 */

send_keys_wait(keys)
char *keys;

  /*
   * Use STREOT and NORESET options for this session.
   * Use STREOT to demonstrate this useful option.  Use
   * NORESET because we are attempting to mimic the
   * operator's actions and he/she did not invisibly enter a
   * RESET key before the current keystroke.
   *
   */
  function = HFUN_SET_SESSION_PARAMETERS;
```

Code Example A-3 Keystroke Playback Program Sample *(11 of 13)*

```
strcpy(data_string, "STREOT NORESET");
data_length = 14;
call_hllapi();
if (return_code != HRC_SUCCESSFUL)
  printf("unable to set session parameters, rc = %d\n",
         return_code);
  exit(1);


/*
 * Send Key to the 3720 PS
 *
 */
function = HFUN_SEND_KEY;
strcpy(data_string, keys);
call_hllapi();

/*
 * Report error and exit if problem detected.
 *
 * Exception: we'll allow the user to get into local input error conditions
 * and get himself out.
 *
 */
if (return_code != HRC_SUCCESSFUL   &&
    return_code != HRC_FUNCTION_INHIBITED )
  printf("unable to send keys %s, rc = %d\n", data_string, return_code);
  exit(1);


/*
 * Wait for the XCLOCK condition to clear.
 *
 */
function = HFUN_WAIT;
call_hllapi();

if (return_code != HRC_SUCCESSFUL && return_code != HRC_FUNCTION_INHIBITED )
  printf("error waiting after sending keys, rc = %d\n", return_code);
  exit(1);

/*
```

Code Example A-3 Keystroke Playback Program Sample *(12 of 13)*

```
   * Reset STREOT to STRLEN
   *
   */
  function = HFUN_SET_SESSION_PARAMETERS;
  strcpy(data_string, "STRLEN");
  data_length = 6;
  call_hllapi();
  if (return_code != HRC_SUCCESSFUL)
    printf("unable to reset session parameters, rc = %d\n",
           return_code);
    exit(1);




/*
 * Call HLLAPI
 *
 * Invoke hllc() and handle system error conditions.
 *
 */

static void
call_hllapi()


  /*
   * Call hllc() with external parameters.
   *
   */
  hllc(&function, data_string, &data_length, &return_code);

  /*
   * Handle system errors.
   *
   */
  if (return_code == HRC_SYSTEM_ERROR
      && function != HFUN_CONVERT_POSITION_OR_ROWCOL)

    printf("System error reported for function %d\n", function);
```

Code Example A-3 Keystroke Playback Program Sample *(13 of 13)*

```
function = HFUN_QUERY_SYSTEM;
data_length = 35;
hllc(&function, data_string, &data_length, &return_code);

printf("Query system values:
printf("EHLLAPI version number = %c\n", data_string[0]);
printf("EHLLAPI level number = %c%c\n", data_string[1], data_string[2]);
printf("EHLLAPI date = %c%c/%c%c/%c%c\n",
                   data_string[3], data_string[4],
                   data_string[5], data_string[6],
                   data_string[7], data_string[8]);
printf("3270 version = %c%c\n", data_string[16], data_string[17]);
printf("hardware = %c\n", data_string[12]);
printf("error code 1 = 0x%c%c%c%c\n",
     data_string[19], data_string[20], data_string[21], data_string[22]);
printf("error code 2 = 0x%c%c%c%c\n",
     data_string[23], data_string[24], data_string[25], data_string[26]);
exit(1);
```

A

Include File

This appendix contains the EHLLAPI include file.

Code Example B-1 `ehllapi.h` Include File Source *(1 of 4)*

```
/*
 * COPYRIGHT (c) 1990 BY BRIXTON SYSTEMS, CAMBRIDGE, MA.
 *
 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
 * COPIED ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH
 * THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR
 * ANY OTHER COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE
 * AVAILABLE TO ANY OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE
 * PROGRAM IS HEREBY TRANSFERRED.
 *
 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT
 * NOTICE AND SHOULD NOT BE CONSIDERED AS A COMMITMENT BY BRIXTON
 * SYSTEMS.
 */

/*
 * ehllapi.h
 *
 * @(#)ehllapi.h     1.2 12/14/92
 *
 */
```

Code Example B-1 `ehllapi.h` Include File Source *(2 of 4)*

```
#ifndef EHLLAPI_H

#define EHLLAPI_H

/*
 * EHLLAPI Function codes
 *
 */

#define  HFUN_CONNECT_PS                1
#define  HFUN_DISCONNECT_PS             2
#define  HFUN_SEND_KEY                  3
#define  HFUN_WAIT                      4
#define  HFUN_COPY_PS                   5
#define  HFUN_SEARCH_PS                 6
#define  HFUN_QUERY_CURSOR_LOCATION     7
#define  HFUN_COPY_PS_TO_STRING         8
#define  HFUN_SET_SESSION_PARAMETERS    9
#define  HFUN_QUERY_SESSIONS            10
#define  HFUN_RESERVE                   11
#define  HFUN_RELEASE                   12
#define  HFUN_COPY_OIA                  13
#define  HFUN_QUERY_FIELD_ATTRIBUTE     14
#define  HFUN_COPY_STRING_TO_PS         15
#define  HFUN_PAUSE                     18
#define  HFUN_QUERY_SYSTEM              20
#define  HFUN_RESET_SYSTEM              21
#define  HFUN_QUERY_SESSION_STATUS      22
#define  HFUN_START_HOST_NOTIFICATION   23
#define  HFUN_QUERY_HOST_UPDATE         24
#define  HFUN_STOP_HOST_NOTIFICATION    25
#define  HFUN_SEARCH_FIELD              30
#define  HFUN_FIND_FIELD_POSITION       31
#define  HFUN_FIND_FIELD_LENGTH         32
#define  HFUN_COPY_STRING_TO_FIELD      33
#define  HFUN_COPY_FIELD_TO_STRING      34
#define  HFUN_SET_CURSOR                40
#define  HFUN_START_KEYSTROKE_INTERCEPT 50
#define  HFUN_GET_KEY                   51
#define  HFUN_POST_INTERCEPT_STATUS     52
#define  HFUN_STOP_KEYSTROKE_INTERCEPT  53
#define  HFUN_SEND_FILE                 90
```

Code Example B-1 `ehllapi.h` Include File Source *(3 of 4)*

```
#define  HFUN_RECEIVE_FILE               91
#define  HFUN_CONVERT_POSITION_OR_ROWCOL 99


/*
 * EHLLAPI Return codes
 *
 */

#define  HRC_SUCCESSFUL                  0
#define  HRC_NO_UPDATES                  0
#define  HRC_PS_ID_INVALID               1
#define  HRC_PARAMETER_ERROR             2
#define  HRC_INVALID_FUNCTION            2
#define  HRC_FT_COMPLETE                 3
#define  HRC_FT_COMPLETE_SEGMENTED       4
#define  HRC_PS_BUSY                     4
#define  HRC_FUNCTION_INHIBITED          5
#define  HRC_DATA_ERROR                  6
#define  HRC_PS_POSITION_INVALID         7
#define  HRC_PROCEDURE_ERROR             8
#define  HRC_SYSTEM_ERROR                9
#define  HRC_FUNCTION_UNAVAILABLE        10
#define  HRC_RESOURCE_UNAVAILABLE        11
#define  HRC_UNDEFINED_COMBINATION       20
#define  HRC_OIA_UPDATED                 21
#define  HRC_PS_ONLY_UPDATED             22
#define  HRC_PS_OIA_UPDATED              23
#define  HRC_STRING_NOT_FOUND            24
#define  HRC_PS_UNFORMATTED              24
#define  HRC_KEYSTROKES_UNAVAILABLE      25
#define  HRC_PS_UPDATED                  26
#define  HRC_FT_TERMINATED               27
#define  HRC_FIELD_ZERO_LENGTH           28
#define  HRC_KEYSTROKES_LOST             31

/*
 * EHLLAPI Convert Postion or RowCol Return codes
 *
 */

#define  HRC_BAD_PARAMETER_99            0
```

Code Example B-1 `ehllapi.h` Include File Source *(4 of 4)*

```
#define  HRC_PS_ID_INVALID_99            9998
#define  HRC_PARAMETER_ERROR_99          9999

#endif EHLLAPI_H
```

HLI Compatibility Functions C

This appendix contains additional functions that are necessary for the EHLLAPI interface to support SunLink HLI.

C.1 set_sna_host Function

This function sets the default SNA host name to the string pointed to by `host` for use with subsequent `start_session` calls. You can change the `host` name between subsequent `start_session` calls to permit connection to multiple gateways. This call has no effect on a running session.

This call is used whether or not the host is actually connected via an SNA link. The link can be SNA, BSC, Local, or TN3270.

This function uses `putenv` to set the SNAHOST environment variable from the user argument. Alternatively, the SNAHOST environment variable can be preset for the application program environment. You must use one of these methods before calling the `start_session` function.

Input:

host	Pointer to the host name string
Output:	N/A

Return codes:

`HLI_OK`	The `host` name is set (only possible return code)

Side effects:	The `host` name can also be set from the SNAHOST environment variable at initialization time. There is no default. Failure to set the `host` name causes subsequent failures in the library.
Notes:	This function is externally known.

C.2 set-model Function

This function sets the model number to the value in `model`. Subsequent session will be of that model number until the next call to `set_model`. This command has no effect on a running session, it is used only to supply default run-time parameters to function internal to the `start_session` call.

Input:

model	Value of the model number of the 3278 being emulated. The valid range is 2 through 5 with a default of 2.
Output:	N/A

Return codes:

`HLI_OK`	The model number was set.
`HLI_ERROR`	The model number was not valid and has not been set.
Side effects:	None
Notes:	This function is externally known.

C.3 start_session Function

This function starts a *new* session with the default SNA host (set by `set_sna_host` or by the environment variable (SNAHOST) by starting a new `td3278` process. Use the `id` parameter is to specify an arbitrary name for this new session. In HLI native mode, the name can be any ASCII character string up to 255 characters in length. In EHLLAPI (`hllc`) mode, the name must be a single ASCII character. Upper and lower case letters are significant. If the name you specify is already in use, you will get an error. Use the show parameter to indicate whether a monitoring screen should be started. Legal values for show are: `SHOWSREEN` and `NOSHOWSCREEN`.

This function performs the following:

- Access the SNAHOST environment variable to determine the host name for connection. Also access the terminal model type variable.
- Determine an available presentation space by using `query_sessions` to obtain a list of currently active presentation spaces. These will be specified by single ASCII characters in the ranges A-Z, a-z, or 0-9. An entry must exist in the `/etc/services` having the prefix `sunehllapi` and ending with one of these ASCII characters for each presentation space. Normally, this will have been done by the installation process.
- Use the above data to start a `sun3270tty` session with the `-g` option to suppress an actual terminal display.
- Next, perform `connect_ps` using the presentation space determined above. Unless overridden by a subsequent `attach` function or `start_session` this becomes the current session. As long as this session is the current session, it will not be necessary to refer to it by an explicit session name in subsequent `hllc` function calls.
- Create a table entry matching the name specified as argument by the user to the actual presentation space parameters. This table will be referenced by the `attach` function.

EHLLAPI function implemented: 1: CONNECT PRESENTATION SPACE

Input:

`id`	Pointer to a non-empty character string of up to 255 bytes used to name the session between a `td3278` process and the SNA host.
`show`	If non-zero, start an `hliscope` window for this session.
Output:	N/A

Return codes:

`HLI_OK`	The session was started successfully.
`HLI_ERROR`	No host name was specified for the session.
`HLI_UNAVAIL`	Resources are not available to start the session.

Side effects:	The `td3278` daemon creates a log file in the current directory, using the following format: `td.<session_name>.log`, where `<session_name>` is the name of the P/S. All diagnostic output is written to this file.
Notes:	This function is externally known.

C.4 *attach* Function

Use this function to connect to an already running session. The named session becomes the current session. You can use this function to switch between multiple *active* host sessions. If you only have one session, you do not need this function.

The name you use must be the same name specified by the `id` parameter to the `start_session` call. When this call completes successfully, `name` becomes the current session for all subsequent function calls.

This function uses the passed argument to find the matching presentation space and set it to be the current session. The argument must be the symbolic name supplied by the user in a previous `start_session` function. All subsequent `hllc` function calls will now reference this session until overridden by another use of `attach` or `start_session`.

EHLLAPI function implemented: 1: CONNECT PRESENTATION SPACE

Input:

model	Pointer to the name of the session (max 255 characters). `name` must be exactly one character to maintain compatibility with EHLLAPI.
Output:	N/A

Return codes:

`HLI_OK`	The named session is now the current session.
`HLI_UNAVAIL`	Unable to find the named session.
Side effects:	The current session does not change if there is an error. If the requested session has failed, this function will complete the termination process and return `HLI_UNAVAIL`.

Notes: This function is externally known.

C.5 close_session Function

This function closes the current session with the host, frees up all memory resources, and kills it associated terminal daemon (`td3278`) process. The IBM host receives a Power-Off message from the gateway process indicating that the virtual "terminal" connected to the LU has been taken out of service.

Note – The caller of the `close_session` function must terminate the host application by sending the appropriate logoff or other action prior to issuing a call to the `close_session` function.

To re-establish the current session, you must issue a `start_session` or `attach` function.

EHLLAPI function implemented: 2: DISCONNECT PRESENTATION SPACE

Input:	N/A
Output:	N/A
Return codes:	
`HLI_OK`	The current HLI session was terminated successfully.
`HLI_CONNECT_ERROR`	There is no current session.
Side effects:	None
Notes:	This function is externally known.

C

Index

T

U

W

X