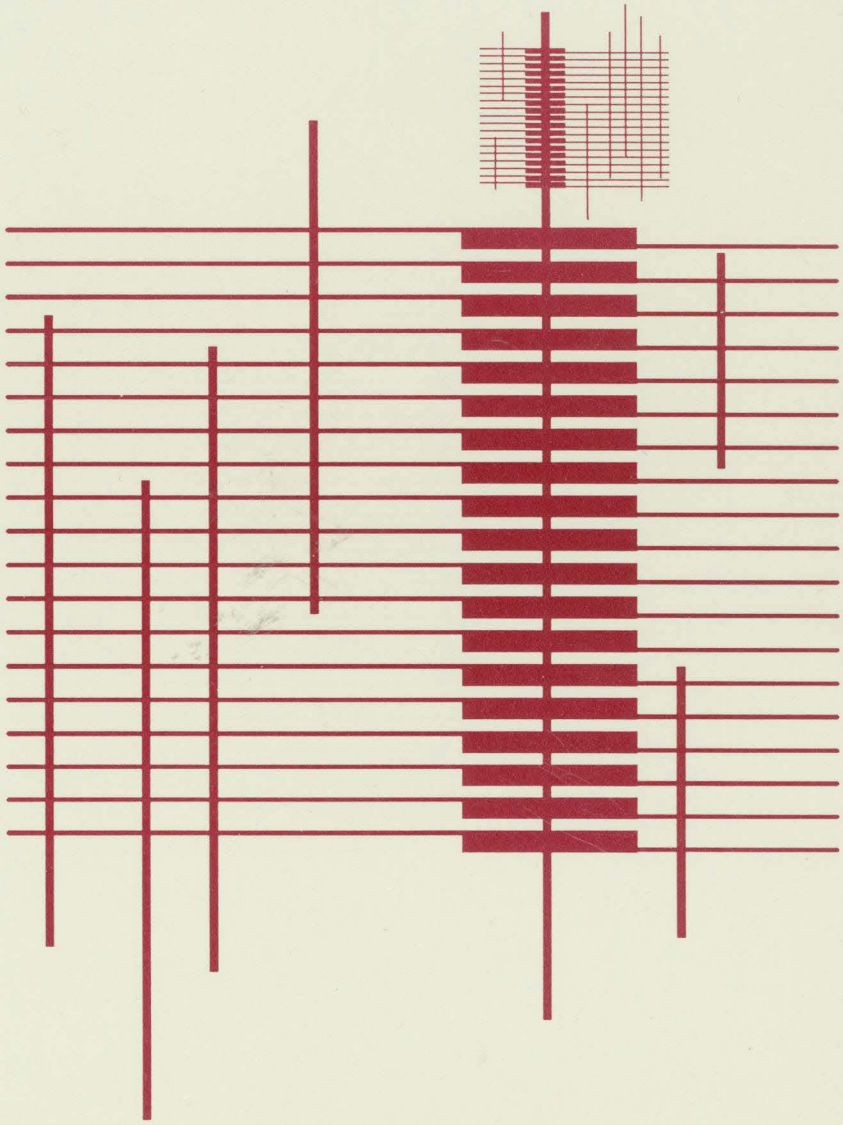
Access API

Programmer's Reference



EICON
TECHNOLOGY

Access API

Access API is a software option for Eicon Technology's Access communications programs. It defines an interface through which third party applications can manipulate the Access program to perform sophisticated communication functions. These functions allow developers and users to take advantage of Eicon Technology hardware and software solutions for micro-to-mainframe connectivity.

Access API supports the full range of Access terminal emulations under TTY, 3270 and 5250 communications. In addition, the API interface for 3270 and 5250 provides support for a variety of third party software.

For 3270 users, Access API is compatible with the IBM PC 3270 Emulation Program Presentation Space API and the IBM PC 3270 Entry Emulator High-Level Language API. It also supports IBM Personal Services/PC, PROFS PC and GDDM-PCLK.

In 5250 setups, Access API provides the facilities for running IBM PC Support/36 and PC Support/38 software.

Access API is available in both stand-alone and LAN versions for the following Eicon Technology software: Access/X.25, Access/QLLC and Access/SDLC.

Second Edition
March 1989

Changes are periodically made to the information in this guide; these changes will be incorporated into new editions of the publication. Eicon Technology may make improvements and/or changes in the products and/or programs described in this publication at any time. A product comment form is provided at the back of this guide. If the form has been removed, please address your comments to: Eicon Technology Corporation 2196 – 32nd Avenue (Lachine), Montreal, Quebec, Canada H8T 3H7.

The Access API
Programmer's Reference

Eicon Technology Corporation

Table of Contents

3270 Applications 11

5250 Applications 12

API Programming 14

Access API Nuts & Bolts 17

Access API Function Calls 21

Executable DOS Files 75

Appendices

Index 112

Overview

What is an API?

API stands for Application Program Interface. As the name suggests, an API defines the interface into an application program. It is the entry point which allows other programs to manipulate an application program the same way a user might.

APIs help facilitate the automation and customization of application functions. Tedious procedures that need to be performed repetitively by the user, such as logging on to several host computers each morning, can be done quickly with an API routine.

What does Access API allow me to do?

Access API allows your application programs to request 3270, 5250 or TTY communication services from Access application software. These services include most of the functions available to users of the Access program, such as: placing calls, receiving calls, file transfers, management of multiple sessions, and various terminal emulations.

The Access API interface lets you customize your communications setup by installing your own programs between local users and outside host computers, or it will allow you to extend your current applications into the X.25 or SNA communication environment.

Access API is extremely flexible. Talk directly to the API interface in assembler, in most high level languages, or from the DOS command line and DOS batch files.

Access API offers more than just a programming interface. It was designed to be compatible with certain IBM protocols in the 3270 and 5250 environments. This means you can combine third party applications with the power of Eicon Technology gateways.

The Access API option for 3270 defines two interfaces. The first is functionally identical to the IBM 3270 Emulation Program (version 3.11), and provides support for IBM Personal Services/PC, GDDM-PCLK and IBM PROFS PC, as well as Tempus-Link and Tempus-Share from Micro Tempus. The second is functionally identical to the IBM PC 3270 Entry Emulator High-Level Language API (EEHLLAPI). Both interfaces allow users to run their current applications over Eicon Technology gateways with no modifications to their code.

For 5250 environments, Access API sports an interface for the IBM PC Support/3X programs that is functionally equivalent to the IBM 5250 Remote Emulation program.

And finally, since Access API is an option to Access communication software, users can still take full advantage of the Access program interactively.

What do I need to know to use Access API?

Many of the API function calls in this manual are similar to commands available to users of Access Application software. To make effective use of these functions you should be familiar with the operation of the Access program, and the specifics of the communications options you will be using (VT100, 3270 or 5250).

Use of the API interface requires an understanding of MS-DOS batch file conventions and 8088 Assembly Language, or any other language that provides access to the PC's registers (such as 'C').

Conventions

References

This manual is intended to be a programmer's reference to Access API, and as such it will not cover information that is already available elsewhere. However, this manual contains references to other documentation which may prove useful. They take the form:

see ☞*This Manual - Appendix D: Reference Books*

EiconCard

The term EiconCard is used to denote any of Eicon Technology's communications controller cards, including: Network Adapter, Dial Network Adapter, Single-Port Communications Coprocessor, Dual-Port Network Adapter, EiconCard, and EiconCard HSI.

Numbers

All numbers in this manual are in decimal unless followed by the suffix 'h', which denotes hexadecimal.

Input

Any command that is entered at the DOS prompt is represented as follows:

Command Line Conventions

GETSCR	The name of the command. You can type this in upper or lower case it does not matter.
/STOP	Items capitalized and in bold type must be entered as shown.
row	Items in lowercase italic type are user supplied input. You should replace these items with the values you need.
[ITEM]	Items enclosed by a pair of square brackets are considered optional. You can either include them or not. Do not type the brackets.
{THIS|THAT}	A vertical line separating two or more items enclosed in braces means make a choice. Enter one of the items when you use the command. Do not type the vertical line or the braces.

How Access API Works

The Big Picture

The Access API communications program works in conjunction with Access software, an EiconCard and its communications software, and your Local Area Network (LAN).

When you install Access API, it attaches itself to the Access application software that resides on your PC. This expands the Access program so it can communicate with other applications. The diagram on the next page illustrates how different programs use the API interface to talk to Access, and how the Access gateway carries out their communications requests. The following paragraphs describe each step in detail.

The API Translation Programs

API5250, IBMAPI and EEHLLAPI are translation programs that come with the Access API package. They allow programs that conform to certain specifications used by IBM in the 3270 and 5250 environments to run without modification on Eicon Technology gateway products. This includes various third-party software, as well as IBM and user developed applications.

IBMAPI and EEHLLAPI mimic interfaces defined by IBM. They accept IBM calls from third-party applications, translate them into Access API commands and pass them to the Access program through the API interface.

API5250 defines an interface that accepts calls from the PC Support/36 and PC Support/38 programs. It translates these calls and passes them to Access for execution.

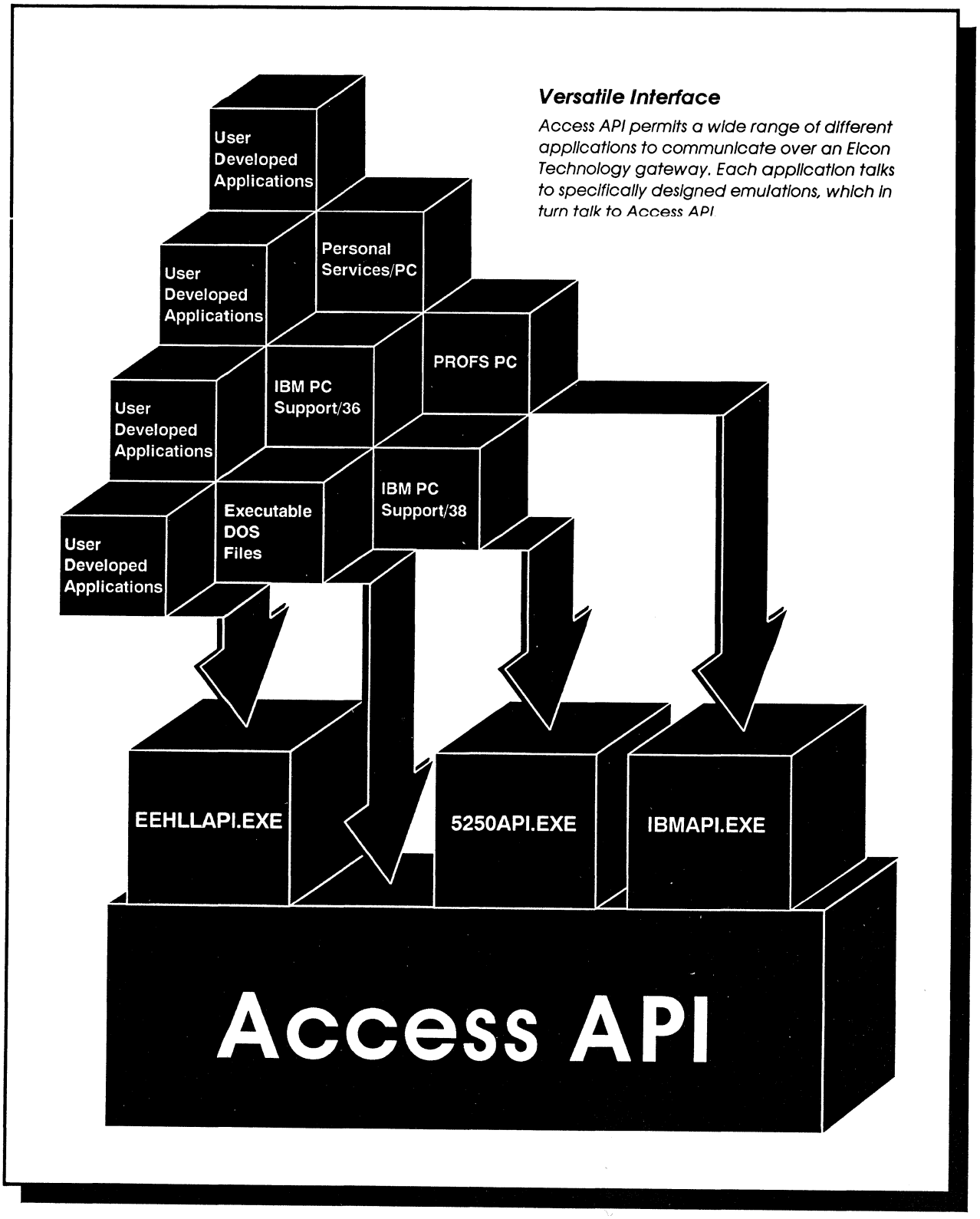
Access and the API Option

When third party application programs request communication services from Access, all data is passed to and from Access through an assembler level interface. This is true for DOS batch files, high level language requests and the IBMAPI translator program.

The API option gives Access the ability to manage this interface. It transfers data to and from third party applications and determines whether input data (API requests) are valid. Some requests are handled internally by the Access program, while others require the use of the EiconCard. The latter reach their destination via the NABIOS and LAN software.

The NABIOS Software

NABIOS is a memory resident program. It manages the link between the Access API program and the EiconCard installed in your PC or on the network gateway.



Versatile Interface

Access API permits a wide range of different applications to communicate over an Elcon Technology gateway. Each application talks to specifically designed emulations, which in turn talk to Access API.

Access API passes commands down to the NABIOS, which in turn routes them to the appropriate software module on the EiconCard. In a stand-alone PC this is straightforward – data is passed on the PC's bus directly to the card. In a LAN environment, the EiconCard will most likely be located in another PC (designated as the communications server or gateway PC). NABIOS on the local PC (or redirector) uses the services of the LAN software to transport data to the gateway PC. NABIOS on the gateway PC then passes this data to the correct module on the EiconCard. Data is returned from the gateway in the same manner.

NABIOS is provided on a diskette that comes with Access Application Software. It is not part of the API package.

The LAN Software

LAN software is the carrier for all data on a local area network. It picks up data from the computers on the LAN, identifies the destination addresses and delivers the information to the proper locations. Access works with all NetBIOS compatible LAN software. Eicon Technology does not sell this software. It is available from a variety of vendors, including: IBM, Novell, 3Com, Banyan and Ungermann-Bass. see ☞ *Access/X.25 User's Guide - Appendix H*

The EiconCard Software

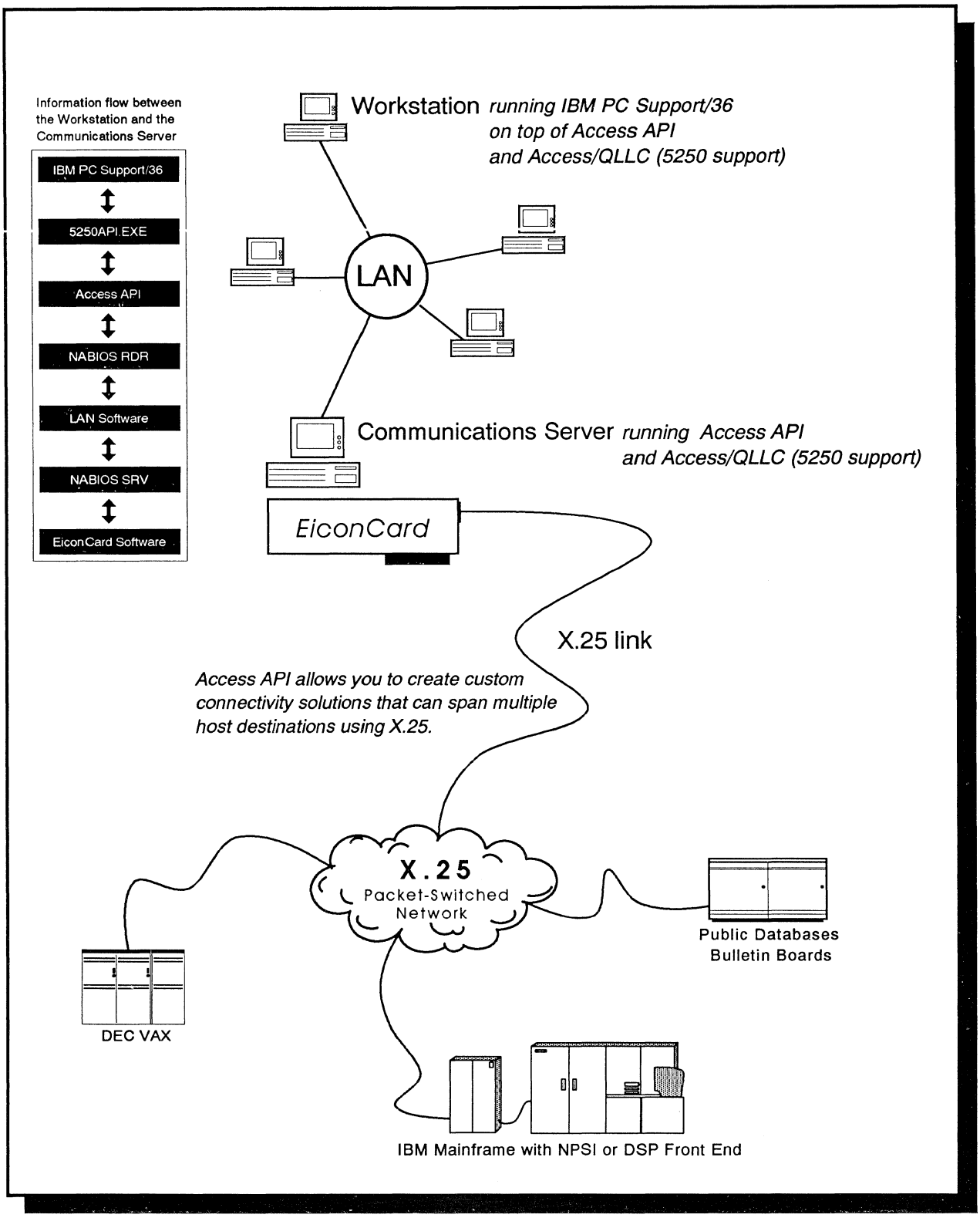
Your EiconCard will contain the software modules that are particular to the communications protocols you are using. Each module is responsible for a specific protocol, be it SNA or X.25, and does all work when data is transmitted or received by the gateway. NABIOS talks directly to the module that will handle the communications task at hand.

The communications modules are loaded into the memory of the EiconCard by a control program (X25NET, SDLC, SNA). Both the communications modules and the control program are part of the Access Application Software package and are not supplied on the Access API diskette(s).

The EiconCard

The EiconCard is the root of the Access gateway. It is an intelligent communications adapter card for the IBM PC or PS/2 family of computers. It has its own on-board processor (Motorola 68008) and memory, and comes with different external interface options to connect with outside lines.

The EiconCard handles all processing of communications protocols leaving the host PC free to perform other tasks.


Information flow between the Workstation and the Communications Server
IBM PC Support/36
5250API.EXE
Access API
NABIOS RDR
LAN Software
NABIOS SRV
EiconCard Software
Workstation running IBM PC Support/36 on top of Access API and Access/QLLC (5250 support)
LAN
Communications Server running Access API and Access/QLLC (5250 support)
EiconCard
X.25 link
Access API allows you to create custom connectivity solutions that can span multiple host destinations using X.25.
X.25 Packet-Switched Network
Public Databases Bulletin Boards
DEC VAX
IBM Mainframe with NPSI or DSP Front End

Installing Access API

File List

Access API is available in three different options: 3270, 5250 or X.25. Depending on the option, different files will be present on the API diskette(s) you receive.

Access API Files

Package	Filename	Description
3270, 5250, X.25	INSTALL.EXE	Installation Program
3270, 5250, X.25	ACCESS.EXE	Access API Program
3270, 5250, X.25	ACCCFG.EXE	Configuration Program
3270, 5250, X.25	ACCESS.CFG	Configuration Datafile
3270, 5250, X.25	ACCESS.FMX	Screen Format File
3270, 5250, X.25	EXECUTE	Directory containing Executable DOS Files
3270, 5250, X.25	SOURCE	Directory containing Source Code
3270	IBMAPI.EXE	IBM/Eicon Technology API translator
3270	SEND.EXE	Emulates the PS/PC Send program
3270	RECEIVE.EXE	Emulates the PS/PC Receive program
3270	EEHLLAPI.EXE	IBM/Eicon Technology API translator
5250	API5250.COM	IBM/Eicon Technology API translator
5250	INTRTR.EXE	Used to interrupt the PC Support/3X router program
5250	DISKS3X.EXE	Used to download PC Support/3X

The Access API program comes on a single 3.5" diskette for PS/2 machines, or up to three double density 5.25" diskettes for XT and AT computers. The diskette labelled *Programming Examples* contains the EXECUTE and SOURCE subdirectories.

Installation Procedure

The INSTALL program will help you set up the API software. In LAN environments you should install the API software on the server (gateway) PC and each PC that will be making API calls. The procedure is as follows:

Insert the API disk (or the disk labelled 1 if you have two) into disk drive A: and enter the INSTALL command at the DOS prompt. If you have two disks, perform the INSTALL commmand on each one.

Install Command

A:INSTALL *drive-name subdirectory-name*

drive-name	identifies the disk where your Access application software is currently installed.
subdirectory-name	is the full path name of the subdirectory

For example the command: INSTALL C: \Access installs the Access API option on the Access software located on drive C: in sub-directory \Access.

The INSTALL program only installs the API option onto the Access program. It will not install any files from the EXECUTE or SOURCE subdirectories, nor will it install the following programs:

IBMAPI.EXE
SEND.EXE
RECEIVE.EXE
EEHLLAPI.EXE
API5250.COM
INTRTR.EXE
DISKS3X.EXE

You should copy these files to the proper directories when you need them. see ☞ *This Manual - 3270 Applications or 5250 Applications*

Configuration

Once INSTALL has finished copying files it will present a new version of the controller card General Configuration Menu. This new menu is identical to the standard Access menu (see ☞ *Access/X.25 User's Guide - Specifying Access Configuration*) except for one addition – a sub-menu under function key F2 which allows you to configure the API software interrupt.

INSTALL copies all configuration information from your previous version of the Access Application software into the new menu. You will not have to redo the settings for any of the Configuration Menus.

Access API Software Interrupt

The default setting for the software interrupt is 7Bh. This can be altered within the range 40h to FFh by reconfiguring the Access program with the ACCESS CONFIG command.

It is not a good idea to set the software interrupt to a value used by other programs. So if you are running any software that already makes use of interrupt 7Bh, change the setting.

Known Software Interrupts

Interrupt Value (Hex)	Software Product
5C	NetBIOS
7A	IBMAPI

note ☞ *Executable DOS files*
If you change the setting of the software interrupt, you must include the /Interrupt parameter each time you use an executable DOS file. This will ensure that the new interrupt is used for API requests.

Alternate Tasks

In order for the API interface to operate properly you should set Enable Alternate Tasks to "Y" in the General Configuration Menu of the Access program.

note ☞ *PC Support/3X and the Access Hot Key*
If you are planning to use IBM PC Support/3X software with Access API you should reconfigure the Hot Key (**see** ☞ *Access/X.25 User's Guide - Specifying Access Configuration). This is necessary because the default setting, ALT-ESC, is used by the PC Support software. If you enter ALT-ESC by accident or you keep it as the "Alternate Task Key," it will lock the keyboard. Enter ALT-ESC again and the keyboard will be unlocked.*

3270 Applications

Access API (3270 Support) allows you to run any program that is compatible with the API defined by IBM for its 3270 Emulation Program Version 3.11, or its Personal Computer 3270 Entry Emulator High-Level Language Version 1.21.

In order to APIs, Access API requires the translation programs IBMAPI.EXE and EEHLLAPI.EXE. These programs convert IBM API and IBM EEHLLAPI calls into Access API calls.

note ☞ *IBMAPI and EEHLLAPI*
Before you make use of IBMAPI.EXE or EEHLLAPI.EXE you will have to copy them to your working directory from the Access API diskette. The INSTALL program does not install them.

Starting Up IBMAPI

Running applications compatible with the IBM 3270 Emulation Program with Access API is easy. First run IBMAPI.EXE by typing the following command at the DOS prompt:

```
IBMAPI
```

Next, start the Access program and make a 3270 call to your host. Depending on the type of IBM API application you will be running, you may have to perform other functions (such as logging onto the host). Once this is done, use the Hot Key or the F4 DOS key to return to DOS. You can now run any application that makes IBM API calls.

note ☞ *Running Personal Services/PC*
SEND.EXE and RECEIVE.EXE allow you to run the IBM Personal Services/PC program. In order for PS/PC to function properly, you must copy SEND.EXE and RECEIVE.EXE into the subdirectory where the PS/PC program is installed. Once this is done you can follow the steps outlined above.

Starting Up EEHLLAPI

Running any IBM EEHLLAPI compatible programs with Access is extremely simple. First, execute EEHLLAPI.EXE by typing the following command at the DOS prompt:

```
EEHLLAPI
```

Next, start the Access program and use the F4 DOS key to return to DOS. Now you can run any EHHLAPPI application and it will use the Access gateway.

note ☞ *EEHLLAPI Programs*
For a complete definition of EEHLLAPI, see IBM document 74X9879, IBM PC 3270 Emulation Program Entry Level Programmer's Guide.

5250 Applications

Access API (5250 Support) allows you to run the IBM PC Support/36 (version 4.0) and PC Support/38 (version 2.0) programs. To do this you will have to make some minor adjustments to certain procedures in your IBM manuals.

Downloading PC Support/3X Software

If you are downloading the PC Support software from a System/3X host, you will need to make some modifications to the directions given in the PC Support/3X Technical Reference manual. The easiest way to look at things is as follows: Running API5250, ACCESS and DISK3X is equivalent to running the IBM Remote Emulation Program with the */s* parameter. So where the PC Support/3X manual mentions the loading of the Remote Emulation Program, substitute the following:

- Run API5250
- Run the Access program
- Call a 5250 session then Hot Key to DOS
- run DISKS3X *s*
  Where s is the name of the selected drive.

note ☞ *Downloading*
The recommended drive value for s is A. It is complicated to use values above C, since the PC's internal switches must be set properly.
Downloading the files in this manner is very slow (1-6 hours). Once the virtual disk software has been downloaded, you can speed things up by using it to download the rest.

Starting Up PC Support/3X

Once again, the easiest way to look at things is as follows: Running API5250 and ACCESS is equivalent to running the IBM Remote Emulation Program. So to start PC Support:

- Run API5250
- Run the Access program
- Call a 5250 session then Hot Key to DOS
- For PC Support/38, run STARTRTR
  For PC Support/36, run BEGINRTR

Interrupting the Router

Use the following in place of "Interrupting the Router" as described in the PC Support/36 and PC Support/38 Technical Reference manuals.

To temporarily interrupt the router, use the INTRTR.EXE program supplied on the Access API diskette(s) as follows:

At the DOS prompt, type:

INTRTR 36 (for PC Support/36)
or
INTRTR 38 (for PC Support/38)

Depending on the type of computer you are connecting to, a host menu will be displayed on your router session screen. For a System/36, this menu is the INQUIRY OPTIONS menu. For a System/38 it will be the SYSTEM REQUEST menu. On either, select option 1 ("Request Command display" on a 36, "Transfer to a secondary inactive job" on a 38). You will now be able to use your PC as a 5250 display station.

The last action performed by INTRTR is to activate the Access program. Therefore it is possible to put your 5250 session on hold and use the other functions of the Access software. The only restriction is that you should not hang up the router session.

Restarting the Router

To restart the router on a System/36, press the emulated command key 1. On a System/38, sign off the secondary interactive job. Both actions will cause the "router screen" to appear. You can now use the Access Hot Key or F4 DOS to return to DOS. It is recommended that you Hot Key to DOS from this screen only. If for some reason the router screen does not appear, the Hot Key will still take you back to DOS. However, this should be used as an emergency exit only. Unpredictable results will occur if you do not return to DOS through the router screen.

note ☞ *Using Virtual Disk or Virtual Printer*
When using virtual disk or virtual printer, Access will temporarily stop processing data intended for other sessions. This is because DOS is not re-entrant. If the other session is a display or printer on the same controller (PU), this could cause the X25 or SDLC window to fill up. If this happens, all communication for that controller will block. There is no recovery except to restart the PU. The router program may not recover from this, and the PC may have to be restarted as well. Therefore it is strongly advised not to have multiple sessions (especially printers) on a PC which is using virtual disk or printer.

API Programming

Access API allows you to manipulate the Access program in two different ways: from an assembler interface, or through commands typed at the DOS prompt.

The Assembler Level Interface

The Assembler Level Interface is at the root of the Access API system. All communication with third-party applications occurs here.

The Assembler Interface consists of a number of different functions. Each function is identified by a unique Service Number. To invoke an API function, you load register AH with the Service Number and supply the necessary data in the Pc's other registers. Next you issue a software interrupt. This activates an Access API routine which processes the function request. Any output from the function call is returned in the Pc's registers.

A complete description of each Access API function call is presented under *Access API Function Calls* starting on page 21 of this manual.

Assembler Functions

Control	Control functions are used to manage the API interface and the Access program.
Session	These functions operate across all communication sessions: 3270, 5250 or TTY. They allow you to establish and manipulate the connections.
3270, 5250, TTY	These three sets of functions are specific to particular types of sessions. They give you control over file transfers, cursor positioning, and various emulation-specific features. The presence of each group of these functions is dependant on the Access API option you are using.

The Executable DOS Files

The Executable DOS files allow users to manipulate Access from the DOS command line or through batch files. They provide nearly all the functionality of the Assembler Interface, although certain assembler commands are now options rather than stand-alone functions.

To invoke API functions from the DOS prompt or from within DOS batch files you simply enter the name of the function followed by the correct parameters.

Each Executable DOS file returns a value upon completion. These values let you know if the function completed successfully or encountered an error. You can make use of these values in batch files by using the DOS ERRORLEVEL command.

see ☞ *This Manual – Appendix C*

A complete description of each Executable DOS file is presented under *Executable DOS Files* starting on page 75 of this manual.

The API Software Interrupt

The Executable DOS files have been coded to use the default software interrupt 7Bh. If you have changed the interrupt, you must use the /I=*Interrupt* flag (where *Interrupt* is the hexadecimal number of the new interrupt setting) with each call to indicate the value of the new interrupt setting.

The Executable files are located in the EXECUTE subdirectory on the API diskette(s). They were created using the Microsoft C Compiler. The C code for each of the modules is located in the SOURCE directory. You can study and modify these files to suit your own needs.

see ☞ *This Manual – Appendix B*

Starting Up the API

To activate the Applications Program Interface, start the Access program and use the Hot Key or the F4 DOS key to put it into the background. When you are back on the DOS command line you can run your API application, or make use of the Executable DOS commands.

The Assembler Level Interface

To activate the interface your first call must be *Service 00: Open API Interface*. Access API will respond with an error message to all other functions until this call is issued.

The Executable DOS Files

Each of these files takes care of opening the API Interface before it executes. There is no special start up procedure.

Closing Down the API

Shutting down involves three steps: closing the API interface, terminating your application, and terminating Access API.

The Assembler Level Interface

Close the Access API interface with *Service 14: Close API Interface*. This will re-enable the user hot key and allow the user to switch into Access and also shut it down.

As long as the API interface is open, the Access API program cannot be terminated. Any attempt to do so will produce the error message "API Active, Cannot Exit".

Executable DOS Files

Each Executable file closes the API interface when it terminates. The API interface is only open for the time it takes the command to run. However, in cases where a function is aborted during execution, you should run ENDAPI.EXE. This will make sure the API interface is correctly closed.

note ☞*Terminating Access*

Access cannot be terminated if another program is active above it in memory. This is done to prevent DOS from crashing.

Access API is a memory resident program. When it terminates, it frees up all the memory it was using. If there is a program above it in memory (a program loaded after Access) a hole will be left in the PC's memory space. When DOS runs into this hole, it crashes the system.

If you cannot terminate all your applications before shutting down Access, then load them before you load Access.

Access API Nuts & Bolts

Registers

Access API treats the PC's registers as follows:

Value not preserved	ax, bx, cx, dx, di
Value preserved	cs, ds, ss, sp, es, bp, si sp

Certain registers perform similar roles across most of the API functions.

Input

Register	Function
AH	service number of the API Function
AL	flags interpreted by the function
BX	session id
CX	length of input string/buffer or keystroke id number
CL	field number byte
ES	segment of input/output buffer/string
DI	offset of input/output buffer/string
DX	offset into device buffer
DL	column
DH	row

Output

Register	Function
AL	return code
BX	session id
DX	offset into device buffer
DL	column
DH	row
CL	character or field number
CH	attribute or number of input fields
CX	status word

Memory Requirements

Access is a memory resident program, therefore it reserves all the memory it requires for proper operation at load time. Each session you define under ACCESS CONFIG will add to the amount of memory Access reserves.

You can get a good idea of how much memory Access requires by using the following table. Total up the memory for each configured emulation and add this to the size of the ACCESS.EXE file.

Memory Requirements (Kbytes)

Emulation	Standard	Extended
ITI or VT100	06	
3278, 3279 Model 2	9.5	11.5
3278, 3279 Model 3	11.5	14
3278 Model 4	13	16.5
3278 Model 5	13.5	17
3270 printers	12	
5250 (all models)	10	
5250 printer	10	

Extended refers to sessions with extended color or high-lighting enabled. If Program Symbols are enabled add 25.5K bytes.

Screen Buffers

Access API keeps a separate buffer for each active session. This buffer reflects the appearance of the session screen. If the session is active in the foreground with screen updates enabled, then any character placed into this buffer will appear on the screen. If the session is in the background, then only the buffer is updated. These buffers enable you to have multiple sessions active without losing data.

API function calls allow you to operate directly on these buffers, so it is important to know their structure. Character positions are referred to by offset, with line 1 on the screen having character positions from offset 0 to 79. Line 2 follows with offsets from 80 to 159, and so on.

note ☞ *Cursor Positions*
Cursor positions are referred to by row and column numbers rather than offset.

There are three types of Screen buffers:

3270 Buffer

This buffer contains an image of the 3270 screen. Characters and Attributes are stored in Device Buffer Format.

see ☞ *IBM PC 3270 Emulation Program User's Guide – Appendix A (Character Code Table)*

3270 Buffer

Terminal	Rows	Columns	Coordinates
3278 Model 2	24	80	00 to 1919
3278 Model 3	32	80	00 to 2559
3278 Model 4	43	80	00 to 3439
3278 Model 5	27	132	00 to 2559
3279 Model 2	24	80	00 to 1919
3279 Model 3	32	80	00 to 2559

5250 Buffer

This buffer contains an image of the 5250 screen. Characters and Attributes are stored in EBCDIC format.

5250 Buffer

Terminal	Rows	Columns	Coordinates
5251 Model 11	24	80	00 to 1919
5251 Model 12	24	80	00 to 1919
5291 Model 1	24	80	00 to 1919
5291 Model 2	24	80	00 to 1919
5292 Model 1	24	80	00 to 1919
3180	21	132	00 to 2559

TTY Buffer

This buffer contains an image of an ITI or VT100 screen. All characters are stored in ASCII format, one character per byte.

TTY Buffer

Terminal	Rows	Columns	Coordinates
VT100	24	80	00 to 1919

Display Adapters

The Access program is aware of the type of display adapter that is installed in a system, be it CGA, EGA, VGA or monochrome. If the program is being used interactively by a user, video modes are saved and restored when the user Hot Keys out of Access into DOS or another application. When manipulating sessions with Access API function calls the responsibility for setting the screen mode rests with the programmer. When a session is brought to the foreground with an API function call, it is assumed that the correct video mode is active. Garbage will result on the screen if the programmer has not set the display to match the requirements of the session.

Device Drivers

If an application program is installed as a DOS Device Driver or is running in the background (activated from an interrupt), then certain restrictions are placed on what it can do.

- only one Access session can be active at any one time. Attempts to open more than one session will result in communications being blocked.
- the API interface must be opened each time the driver is called and closed before the driver returns. This means using *Service 00h: Open API Interface* at the start of your driver, and *Service 14h: Close API Interface* at the end.
- the driver may not initiate a session with *Service 01: Call* or *Service 17h: Answer*.
- it is prohibited to use *Service 13h: Hot Key Switch* to give control to the user or to access a session that already has a file transfer operation in progress.
- it is also prohibited to use any API functions that will result in Access API making DOS function calls. This includes:

3270 Functions	Service 12h: File Transfer Service 15h: Save Screen
5250 Functions	Service 3Dh: Save Screen
TTY Functions	Service 1Dh: File Transfer Service 1Eh: File Capture

Access API Function Calls

The following section presents a detailed description of each of the Access API assembly language function calls.

Divisions

There are 48 Access API function calls. Each function is classified in one of five divisions: Control Function, Session Function, 3270 Function, 5250 Function or TTY Function. Control and Session functions can be used with any type of session. 3270, 5250 and TTY functions are specific to their respective session types.

Service 14h: Close API Interface

Control Function

Input

AH = 14h
AL = 00

Output

none

Description

This function ends API activity and reenables the user Hot Key. If Close API Interface is omitted, the ability to bring up Access through the user Hot Key will be lost. In this case, to return to Access the user should type EXIT at the DOS prompt.

Note

This function does not terminate the Access program. Access and any active sessions will still be available for use.

Service 13h: Hot Key Switch

Control Function

Input

AH = 13h
AL = type of switch 00h – regular mode
01h – enhanced mode
ES:DI = address of the Key List (enhanced mode only)

Output

AX = Exit Key

Description

This service causes the calling application to be suspended and activates the Access program in interactive mode. The current foreground session is displayed with keyboard input and screen updates enabled.

Regular Mode

If the user attempts to exit from Access using F10 QUIT, the message "API Active, Cannot Exit" is displayed. All other Access functions can be executed. Control is returned to the application only when the user strikes the F4 DOS key.

Enhanced Mode

This mode allows the calling application to put restrictions on exactly what a user can do when placed back into Access. Enchanced Mode disables all Access function keys not specific to an emulation. This includes F1 Directory, F2 Call, F3 Answer, F4 DOS F6 Switch, F10 Exit. In addition, wherever the F4 DOS (or ALT-ESC hotkey) was active, Access now checks each user keystroke against the Key List for a match. If a match is found Access returns to DOS with the DOS ScanCode value of this key as Exit Key.

Key List

Byte	Description
0	Number of ScanCodes in the Key List
2–	2 bytes for the ScanCode of each key

Note

This function will only work if "Enable Alternate Tasks" is set to "Y" under ACCESS CONFIG. If set to "N" this function will execute but nothing will happen. Also, since actual ScanCodes are used a conflict with a configured key is possible. This function takes precedence.

Service 00h: Open API Interface

Control Function

Input

AH = 00h
AL = 00

Output

AL = Return Code 00h – successful

Description

This function must be the first API call issued. It activates the API interface and disables the user Hot Key (ALT - ESC) for the duration of API activity (Assembler Service *13h: Hot Key Switch* is the only exception to this). Access screen updates are suspended for any active sessions.

Service 0Fh: Query Access Type

Control Function

Input

AH = 0Fh
AL = 00

Output

CX = Access software Option ID
DX = Access software Version Number

Description

This function allows an application to determine the Option ID and the Version Number of the Access software it has contacted.

Access Software	Option ID
QLLC	102
SDLC	106
X25	101

Service 17h: Answer

Session Function

Input

AH = 17h
AL = 00
AL = Mode — 00h – answer by name; 01h – answer by values
CX = length of Answer Data in bytes
ES:DI = address of Answer Data

Output

AH = Return Code — 00h – successful; 01h – name not found or invalid call values; 02h – unsuccessful
BX = Session ID if successful

Description

This function sets up an ITI answer session. It returns a Session ID immediately without waiting for completion of the connection. Use *Service 11: Query Session Status* to determine when the incoming call has connected.

This function is for ITI session only. It operates in two modes.

Answer by Name

In this mode, Access API tries to match the character string supplied as Answer Data with the names in the current Access Calling Directory. If a match is found (the match operation is not case sensitive), Access creates an Answer session using the parameters for the entry.

Answer by Values

In this mode, an answer session can be created without refering to an Access Calling Directory entry. Instead of containing a Calling Directory Name, Answer Data should contain a series of fields that define the parameters of such an entry. Each field contains three items: a Field Identifier byte, a Field Length byte and Field Data.

Field Identifiers are codes that represent Access Calling Directory parameters. A list of Field Identifiers is presented on the next page.

Field Length is the length of the Field Data in bytes.

Field Data is a string of characters.

The first word of the Answer Data buffer must contain the total length of the Answer Data buffer. This should be followed by fields for Call Name and Call Type respectively. Additional fields can be specified in any order.

Field Identifiers

Value	Field Name
81h	Call Name
82h	Call Type
83h	DTE Address
84h	Facilities
85h	Call User Data
86h	Parameters
87h	PU Name
88h	LU Number
89h	Printer Model
8Ah	Printer Destination

Example

The following example creates an answer session called "Anyone" using the Call by Values option. It will connect with a call from any DTE address.

Answer Example

Byte	Contents	Description
0 – 1	18	Total length of Call Data
2	81h	Field ID for Call Name
3	6	Length of Call Name
4 – 9	Anyone	Call Name
10	82h	Field ID for Call Type
11	3	Length of Call Type
12 – 14	ITI	Call Type
15	83h	Field ID for DTE Address
16	1	Length of DTE Address
17	*	DTE Address (accept all)

Service 04h: Background

Session Function

Input

AH = 04h
AL = Screen Update Flag — 00h – background; 01h – hold (TTY only)
BX = Session Id

Output

AL = Return Code — 00h – successful; 01h – invalid session id

Description

This service places the session with the specified Session ID into the background and stops all screen updates. For TTY type sessions (ITI, VT-100) the session may be put on Hold instead into the background.

Service 01h: Call

Session Function

Input

AH = 01h
AL = Mode — 00h – call by name; 01h – call by values
CX = length of Call Data in bytes
ES:DI = address of Call Data

Output

AH = Return Code — 00h – successful; 01h – name not found or invalid call values; 02h – call failed
BX = Session ID if call is successful
BH = X.25 Cause Code if call fails
BL = X.25 Diagnostic Code if call fails

Description

This function establishes a 3270, 5250 or ITI connection. It operates in two modes.

Call by Name

In this mode, Access API tries to match the character string supplied as Call Data with the names in the current Access Calling Directory. If a match is found (the match operation is not case sensitive), Access API will try to make the call using the parameters for the found entry. If the call is successful then a Session ID is returned and the session is placed in the background.

If the Call Name is that of an Autolog (.ALG) file, the file will be opened and executed exactly as if it were run under Access. Depending on the instructions in the Autolog file this might lead to screen updates being enabled and messages being written to the display.

see ☞*Access/X.25 User's Guide – Chapter 10: Autostart & Autolog Features*

Call by Values

In this mode, a call can be made without refering to an Access Calling Directory entry. Instead of providing a call name, Call Data should contain a series of fields that define the parameters of a call. Each field contains three items: a Field Identifier byte, a Field Length byte and Field Data.

Field Identifiers are codes that represent Access Calling Directory parameters. A list of Field Identifiers is presented on the next page.

Field Length is the length in bytes of the Field Data.

Field Data is a string of characters.

The first word of the Call Data buffer must contain the total length of the Call Data buffer. This should be followed by fields for Call Name and Call Type respectively. Additional fields can be specified in any order.

Field Identifiers

Value	Field Name
81h	Call Name
82h	Call Type
83h	DTE Address
84h	Facilities
85h	Call User Data
86h	Parameters
87h	PU Name
88h	LU Number
89h	Printer Model
8Ah	Printer Destination

Example

The following example calls Eicon Technology in Montreal using the Call by Values option. It creates a session named "Eicon."

Call Example

Byte	Contents	Description
0 – 1	24	Total length of Call Data
2	81h	Field ID for Call Name
3	5	Length of Call Name
4 – 8	Eicon	Call Name
9	82h	Field ID for Call Type
10	3	Length of Call Type
11 – 13	ITI	Call Type
14	83h	Field ID for DTE Address
15	8	Length of DTE Address
16 – 23	56300026	Eicon's DTE Address

Service 03h: Foreground

Session Function

Inputs

AH = 03h
AL = Screen Update Flag — 00h – do not enable screen updates; 01h – enable screen updates
BX = Session ID

Output

AL = Return Code — 00h – successful; 01h – invalid session id

Description

This function brings the session with the specified Session ID to the foreground. If the Screen Update Flag is set to 1, all session activity will be displayed on the screen. If another session was already in the foreground then that session will automatically be placed into the background.

Note

It is the programmer's responsibility to make sure that the display adapter is in the proper mode (text or graphics) when a session is brought to the foreground. Access API assumes that the adapter is configured according to the requirements of the session. If a graphics session is brought to the foreground and the display is in text mode, the graphic screen will appear translated into random PC characters.

Service 02h: Hang Up

Session Function

Input

AH = 02h
AL = 00
BX = Session ID

Output

AL = Return Code

00h – successful
01h – invalid session id

Description

This function hangs up the session with the specified Session ID. The session must be on-line.

Note

This function will not return until the session has been hung up. If there is a communication problem on the line that hampers the hang up procedure, the function might hold for a long time before completing.

Service 11h: Query Status

Session Function

Input

AH = 11h
AL = Mode
- 00h – regular mode
- 01h – X.25 mode
- 80h – query & clear status line
- 81h – overwrite device status area
- 82h – overwrite help area
- 83h – overwrite device status & help area

BX = Session ID
ES:DI = address of a buffer to receive the Device Information String
ES:SI = address of a buffer to receive the Help String

Output

All Modes supply a Return Code in AL upon completion. Other output however differs between modes.

AL = Return Code
- 00h – successful
- 01h – invalid session id

Regular Mode (00h)

CL = State Byte
- 01h – session in foreground
- 02h – session on hold
- 03h – session in background

CH = Status Byte
- 01h – waiting
- 02h – calling
- 03h – connected
- 04h – hanging up
- 05h – hung up
- 06h – aborted

X.25 Mode (01h)

CL = State Byte
- 01h – normal
- 02h – X.25 reset received
- 03h – X.25 clear received

CH = Status Byte
- 01h – waiting
- 02h – calling
- 03h – connected
- 04h – hanging up
- 05h – hung up
- 06h – aborted

DH = X.25 Cause Code
DL = X.25 Diagnostic Code

Query & Clear Status Line (80h)

Device Status Length and Help Length are returned in the Device Information String.

There are no output values for the following Modes.

Overwrite Device Status Area (81h)

Overwrite Help Area (82h)

Overwrite Device Status & Help Area (83h)

Description

This function allows programs to monitor the status of Access sessions, as well as write information to the Access Status Line displayed at the bottom of each emulation screen.

Modes 01 and 02 return information on Access Sessions. Modes 80h, 81h, 82h and 83h control overwriting of the Status Line.

Regular Mode

This mode functions for all Access sessions. It returns a State byte, a Status Byte, and a 65 byte Device Information String. The Device Information String has the same format as the Device Information displayed at the bottom of the ACCESS offline menu screen.

Device Information String for Modes 00h, 01h

Offset	Description
00	Session name (and LU number if 3270 type)
16	Device type (3270 SDLC, 3720 QLLC, 5250 SDLC, 5250 QLLC, ITI, VT100)
29	PU name or the DTE address

X.25 Mode

This mode returns detailed information about X.25 sessions. It returns Cause and Diagnostic codes, as well as Status and State bytes. The State byte is cleared after the function completes.

see ☞*Access/X.25 User's Guide – Appendix D: X.25 Cause and Diagnostic Codes*

Query & Clear Status Line

This mode returns two words in the Device Information Buffer: Device Status Length and Help Length. It also resets the Staus Line to its default value, cancelling out any previous modifications.

Device Status Length is the number of character spaces available in the Device Status Area of the Status Line. This is the leftmost field which normally shows the *Online to* message.

Help Length is the number of character spaces available in the Help Area of the current session. This is the area where function key assignments are normally displayed. For 3270 and 5250 sessions this is smaller than for than for ITI or VT100, as part of the Status Line is used for the OIA display.

Device Information String for Mode 80h

Offset	Description
00	Device Status Length
16	Help Length

Overwrite Device Status Area

This mode overwrites the Device Status Area on the Status Line with bytes from the Device Information String and resets the Help Area to its default value. The number of bytes written is equal to Device Status Length.

Overwrite Help Area

This mode overwrites the Help Area on the Status Line with bytes from the Help String and resets the Device Status Area to its default value. The number of bytes written is equal to Help Length.

Overwrite Device Status & Help Area

This mode overwrites the Device Status Area with bytes from the Device Information String, and the Help Area with bytes from the Help String.

Service 12h: File Transfer

3270 Function

Input

AH = 12h
AL = file transfer control
- 00h – start file transfer
- 01h – abort file transfer

BX = Session Id
CX = length of File Transfer Command String
ES:DI = address of File Transfer Command String

Output

AL = Return Code
- 00h – successful
- 01h – invalid session id
- 02h – invaiid length
- 03h – input inhibited
- 04h – unable to enter command in field
- 05h – unable to open the file
- 06h – syntax error in file transfer string
- 07h – not allowed to terminate

Description

This function allows applications to control 3270 file transfers. The length of the File Transfer Command String must be 200 bytes or less. It must be in the same format as when using Access interactively to initiate a 3270 file transfer.

Access initiates the file transfer by entering the File Transfer Command String into the 3270 screen of the specific session. The string is placed into the first available unprotected field after the current cursor position. If the file transfer command string is longer than the field, error 04 will occur. If the string is accepted, file transfer is initiated, and the function returns to the application with code 00. The application must use *Service 10: Query Status* to determine when the file transfer has completed.

Service 0Ch: Find Next Attribute

3270 Function

Input

AH = 0Ch
AL = Search Type — 00h – any attribute; 01h – unprotected field attribute
BX = Session ID
DX = Screen Buffer Offset where search will start

Output

AL = Return Code — 00h – successful; 01h – invalid session id; 02h – invalid offset
DX = Offset of Attribute — -1h – if not found

Description

This function finds the next 3270 attribute searching forward from the specified Screen Buffer Offset. To obtain the value of the attribute, use *Service 07: Read Buffer Character*.

Service 0Dh: Find Previous Attribute

3270 Function

Input

AH = 0Dh
AL = search type — 00h – any attribute; 01h – unprotected field attribute
BX = Session ID
DX = Screen Buffer Offset where search will start

Output

AL = Return Code — 00h – successful; 01h – invalid session id; 02h – invalid offset
DX = Offset of Attribute — -1h – if not found

Description

This function finds the previous 3270 attribute searching backward from the specified Screen Buffer Offset. To obtain the value of the attribute, use *Service 07h: Read Buffer Character*.

Service 05h: Query Cursor

3270 Function

Input

AH = 05h
AL = 00
BX = Session ID

Output

AL = Return Code

00h – successful
01h – invalid session id

DL = Cursor Column
DH = Cursor Row

Description

This service returns the position of the cursor in the specified session. Maximum values for Cursor Column and Cursor Row depend on the terminal emulation in use.

Service 10h: Query Status

3270 Function

Input

AH = 10h
AL = 00
BX = Session ID

Output

AL = Return Code
- 00h – successful
- 01h – invalid session id

CX = Status Word
DL = number of columns
DH = number of rows

Description

This function returns a Status Word that allows an application to monitor the activity of API function calls.

Status Word

Bit	Meaning	Notes
0	the screen buffer has been modified	cleared to zero each time this function is called
1	OIA has been modified	cleared to zero each time this function is called
2	the session is connected	
3	a file transfer is in progress	
4	user traffic is allowed	Input Not Inhibited, the device can now respond to the host
5	data traffic is allowed	activated by the host
6	data received from the host	each AID key sent sets this bit to 0, a reponse from host sets it to 1
7	the screen is in graphics mode	
8–A	model number of the terminal	
B	extended highlight and color is active	
C	programmed symbols set active	
D–F	reserved	

Service 07h: Read Buffer Character

3270 Function

Input

AH = 07h
AL = format of character — 00h – device buffer format; 01h – ASCII
BX = Session ID
DX = Screen Buffer Offset

Output

AL = Return Code — 00h – successful; 01h – invalid session id; 02h – invalid offset or format
CL = Character
CH = value for Extended Attribute Data

Description

This service returns the character at the indicated Screen Buffer Offset for the specified session.

Note

If the character is an attribute byte it will be returned in Device Buf-fer Format even if ASCII was specified.

Service 0Eh: Read OIA

3270 Function

Input

AH = 0Eh
AL = 00
BX = Session ID
ES:DI = address of a buffer to receive the OIA String

Output

AL = Return Code

00h – successful
01h – invalid session id

Description

This functions returns a subset of the values for the 3270 Operator Information Area as specified by IBM. The OIA String is 46 bytes in length.

OIA String

Offset	Value
00	System Readiness and Connection Symbols
10	Input Inhibited
22	Reminders
34	Shifts

see ☞*Access/QLLC (3270 Support) User's Guide– Operator Information Area Messages*

Service 0Bh: Read String

3270 Function

Input

AH = 0Bh
AL = format of data — 00h = device buffer format; 01h = ASCII
BX = Session ID
CX = length of the String to read
DX = Screen Buffer Offset
ES:DI = address of the String

Output

AL = Return Code — 00h – successful; 01h – invalid session id; 02h – invalid offset

Description

This service returns a String from the specified 3270 session. If the length of the String forces a read beyond the end of the 3270 Screen Buffer, error 02 occurs.

Service 15h: Save Screen

3270 Function

Input

AH = 15h
AL = 00
BX = Session ID
CX = length of Destination String
ES:DI = address of the Destination String

Output

AL = Return Code

00h – successful
01h – invalid session id
02h – invalid length
05h – unable to open the file

Description

This function writes an image of the specified session's Screen Buffer to the indicated Destination. The Destination String can be the name of any valid DOS file or device. The Destination String cannot exceed 32 bytes. This command is functionally identical to the command under Access.

Service 09h: Send Keystroke

3270 Function

Input

AH = 09h
AL = 00
BX = Session ID
CX = KeyID of keystroke

Output

AL = Return Code

00h – successful
01h – invalid session id
06h – invalid KeyID
10h – input inhibited
12h – successful, AID sent

Description

This function allows application programs to mimic user keyboard entry for a 3270 session. All keys sent by this function will produce the same results as if typed on the keyboard. In addition to regular keys, AID keys such as ENTER or PF1 may be sent.

Note

Applications should use *Service 10h: Query Status* to determine if data can be sent to the host. In an interactive scenario where the host application and the user application exchange a series of screens, the user should wait until *Service 10h* reports bits 4,5 and 6 as 1 before trying to send a keystroke.

Service 06h: Set Cursor

3270 Function

Input

AH = 06h
AL = 00
BX = Session ID
DL = Cursor Column
DH = Cursor Row

Output

AL = return code

00h – successful
01h – invalid session id
02h – invalid input data

Description

This service places the cursor for the specified 3270 session. Maximum values depend on the terminal emulation in use.

Note

Error 02 indicates that the cursor values are out of range or are within a protected field.

Service 08h: Write Buffer Character

3270 Function

Input

AH = 08h
AL = format of character — 00h = device buffer format; 01h = ASCII
BX = Session ID
DX = Screen Buffer Offset
CL = Character

Output

AL = Return Code

00h – successful
01h – invalid session id
02h – invalid offset or format
10h – input inhibited
12h – invalid character
0Eh – protected field

Description

This service places a character into the 3270 Screen Buffer of the specified session. Cursor position is updated.

Service 0Ah: Write String

3270 Function

Input

AH = 0Ah
AL = format of data — 00h = device buffer format; 01h = ASCII
BX = Session ID
CX = length of String
DX = Screen Buffer Offset
ES:DI = address of String

Output

AL = Return Code

- 00h – successful
- 01h – invalid session id
- 02h – invalid offset
- 10h – input inhibited
- 12h – invalid character
- 0Eh – protected field encountered

Description

This function writes a String to the specified location in the Screen Buffer. Cursor position is updated.

Note

If the String overlaps attribute bytes or protected fields, the characters that fall on these positions are skipped.

Service 31h: Query Cursor

5250 Function

Input

AH = 31h
AL = 00
BX = Session ID

Output

AL = Return Code

00h – successful
01h – invalid session id

DL = Cursor Column
DH = Cursor Row

Description

This service returns the current cursor position for the specified 5250 session.

Service 35h: Query Field Data

5250 Function

Input

AH = 35h
AL = format — 00h – EBCDIC; 01h – ASCII
BX = Session ID
CL = Field Number Byte
ES:DI = address of a buffer that will receive the Field Data

Output

AL = Return Code — 00h – successful; 01h – invalid session id; 02h – invalid format or field number

Description

This function returns the data stored in the specified input field. The buffer at ES:DI should be set to the field length value returned by *Service 34: Query 5250 Field Description*.

Note

Applications can determine the Field Number Byte using *Service 33h: Query Field Number Byte*.

Service 34h: Query Field Description

5250 Function

Input

AH = 34h
AL = 00
BX = Session ID
CL = Field Number Byte
ES:DI = address of a buffer to receive the Field Descriptor

Output

AL = Return Code

00h – successful
01h – invalid session id
02h – field number is out of range

Description

This function returns an 8 byte Field Descriptor that lets an application determine various useful information about the fields on a 5250 emulation screen.

Field Descriptor

Offset	Description
00	field location (offset into 5250 buffer)
02	length of field
04	Field Format Word (00 00 for Output fields)
06	Resequence Number (00 for Output fields)
07	Self-Check Module (00 for Output fields)

Note

Applications can determine the Field Number Byte using *Service 33h: Query Field Number Byte*.

Service 33h: Query Field Number

5250 Function

Input

AH = 33h
AL = 00
BX = Session ID
DL = Cursor Column
DH = Cursor Row

Output

AL = return code
- 00h – successful
- 01h – invalid session id
- 02h – cursor position out of range

CL = Field Number Byte
CH = total number of input fields defined

Description

This function allows an application to identify the Field Number Byte of each field on a 5250 screen. Once an application has this information it can use other functions to manipulate the field.

The Field Number Byte contains bits that specify the Field Number and Field Type. Access API assigns a unique Field Number to each field on the 5250 screen. Field Type is either input or output. Any API function call that operates on a 5250 field requires the Field Number Byte as input.

To get the Field Number Byte of a field requires a general method for detecting fields. This function uses a cursor position on the 5250 screen as a starting point. If the specified Cursor Row and Cursor Column fall on an input field then bit 7 of the Field Number Byte is set to 0, and bits 0–6 give the Field Number. If Cursor Row and Cursor Column fall on an output field then bit 7 is set to 1 and bits 0–6 give the number of the preceding input field. If there is no preceding input field then bits 0-6 are set to zero.

Service 3Bh: Query DIM

5250 Function

Input

AH = 3Bh
AL = 00
BX = Session ID
ES:DI = address of a buffer to receive the OIA String

Output

AL = Return Code

00h – successful
01h – invalid session id

Description

This service returns a 32 byte DIM String. The DIM (Display Indicator Messages) String contains the messages displayed by Access in the OFFLINE and ONLINE menus.

see ☞*Access/SDLC (5250 Support) or Access/QLLC (5250 Support) User's Guide – Appendix B.*

Service 3Ch: Query Status

5250 Function

Input

AH = 3Ch
AL = 00
BX = Session ID

Output

AL = Return Code

00h – successful
01h – invalid session id

CX = Status Word

Description

This function returns a Status Word that allows an application to monitor 5250 sessions.

Status Word

Bit	Meaning	Notes
0	message waiting	
1	data link active	
2	SS-LU session active	
3	LU-LU session active	
4	cursor position modified	cleared to 0 after this function returns
5	format table modified	cleared to 0 after this function returns
6	status line modified	cleared to 0 after this function returns
7	5250 device buffer modified	cleared to 0 after this function returns
8-9	reserved	
10	error line in use	
11	input inhibited was turned off	cleared to 0 after this function returns
12	input inhibited was turned on	cleared to 0 after this function returns
13	input inhibited	
14	command mode	
15	insert mode	

Service 37h: Read Buffer Character

5250 Function

Input

AH = 37h
AL = format — 00h – EBCDIC; 01h – ASCII
BX = Session ID
DL = Column
DH = Row

Output

AL = Return Code — 00h – successful; 01h – invalid session id; 02h – invalid format or row/column outside of screen
CL = Character

Description

This service returns the character at the specified position in the 5250 Screen Buffer.

Service 39h: Read String

5250 Function

Input

AH = 39h
AL = format — 00h – EBCDIC; 01h – ASCII
BX = Session ID
CX = length of String
DL = Column
DH = Row
ES:DI = address of buffer to receive the String

Output

AL = Return Code
- 00h – successful
- 01h – invalid session id
- 02h – invalid format or string is (partially) outside of screen

Description

This function returns a string of characters from the 5250 Screen Buffer. It does not recognize input or output fields, and will return a value right across field boundaries.

Service 3Dh: Save Screen

5250 Function

Input

AH = 3Dh
AL = 00
BX = Session ID
CX = length of Destination String
ES:DI = address of Destination String

Output

AL = return code

00h – successful
01h – invalid session id
02h – invalid string length
05h – unable to open the file

Description

This function copies the contents of the 5250 Screen Buffer to the file or device specified by the Destination String. The Destination String must not exceed 32 bytes.

Service 30h: Send Keystroke

5250 Function

Input

AH = 30h
AL = 00
BX = Session ID
CX = KeyID of keystroke

Output

AL = Return Code

00h – successful
01h – invalid session id
10h – input inhibited
11h – key results in an operator error
14h – next alphanumeric key will produce an operator error

Description

This function allows an application program to mimic user keyboard entry for a 5250 session. All keys sent by this function produce the same results as if typed on the keyboard. In addition to regular keys, AID keys such as ENTER or PF1 may be sent.

Notes

Before using this function, applications should determine if keyboard entry is allowed by calling *Service 3Ch: Query 5250 Status*.

Service 32h: Set Cursor

5250 Function

Input

AH = 32h
AL = 00
BX = Session ID
DL = column
DH = row

Output

AL = return code

00h – successful
01h – invalid session id
02h – cursor position out of screen
10h – move ignored because input inhibited
11h – move results in an operator error

Description

This service sets the cursor position for the specified 5250 session.

Service 38h: Write Buffer Character

5250 Function

Input

AH – 38h
AL – format

- 00h – EBCDIC
- 01h – ASCII

BX – Session ID
CL – Character
DL – Column
DH – Row

Output

AL – Return Code

- 00h – successful
- 01h – invalid session id
- 02h – invalid format or illegal row/column position
- 10h – ignored - input inhibited
- 11h – operator error
- 14h – executed, but a warning is issued that the next alphanumeric key, if sent, will produce an operator error

Description

This service places a Character into the 5250 buffer at the specified position.

Service 3Ah: Write String

5250 Function

Input

AH = 3Ah
AL = format

- 00h – EBCDIC
- 01h – ASCII

BX = Session ID
CX = length of String
DL = Column
DH = Row
ES:DI = address of String

Output

AL = Return Code

- 00h – successful
- 01h – invalid session id
- 02h – invalid format or row/column position outside of screen
- 10h – ignored - input inhibited
- 11h – operator error
- 14h – executed, but a warning is issued that the next alphanumeric key, if sent, will produce an operator error

Description

This service writes a String to the Screen Buffer of the specified session. Characters are placed mimicking keyboard entry (skip to next field when a field becomes full, wrap around at the end of the screen).

Service 36h: Write String to Field

5250 Function

Input

AH = 36h
AL = format

- 00h – EBCDIC
- 01h – ASCII

BX = Session ID
CL = Field Number Byte
ES:DI = address of String

Output

AL = Return Code

- 00h – successful
- 01h – invalid session id
- 02h – invalid format or field number out of range
- 10h – ignored - input inhibited
- 11h – operator error
- 14h – executed, but a warning is issued that the next alphanumeric key, if sent, will produce an operator error

Description

This service writes a String into the specified 5250 field.

Note

Characters are placed into the field one at a time, mimicking keyboard entry. If illegal characters are contained in the string an operator error is issued when they are encountered. Previous characters are still entered into the field. Strings that exceed the field size are truncated.

Applications can determine the Field Number Byte using *Service 33h: Query Field Number Byte*.

Service 1Eh: File Capture

TTY Function

Input

AH = 1Eh
AL = file capture control — 00h – capture on; 01h – capture off
BX = Session ID
CX = length of Device Name
ES:DI = address of Device Name

Output

AL = return code — 00h – successful; 01h – invalid session id; 02h – invalid length; 05h – unable to open file

Description

This function toggles file capture mode on or off. The captured file can be redirected to any legal DOS device (printer, file, screen).

see ☞ *Access/X.25 User's Guide – The F2 Capture Command*

Service 1Dh: File Transfer

TTY Function

Input

AH = 1Dh
AL = file transfer control
- 00h – send binary file
- 01h – send ASCII file
- 02h – abort file send

BX = Session ID
CX = length of File Name
ES:DI = address of File Name

Output

AL = Return Code
- 00h – successful
- 01h – invalid session id
- 02h – invalid length
- 05h – unable to open file
- 06h – file transfer in progress

Description

This function allows applications to control TTY File Transfers. The File Name should not exceed 32 bytes. Applications should use *Service 1Fh: Query TTY Status* to determine when the file send is complete.

Service 22h: Flush

TTY Function

Input

AH = 22h
AL = 00
BX = Session ID

Output

AL = Return Code

00h – successful
01h – invalid session id

Description

This function can be used to force data in the PAD buffer to be transmitted.

Notes

It may not be necessary to call this function if the PAD parameters are set to transmit the data automatically.

see ☞*Access/X.25 User's Guide – Appendix B. ITI PAD Parameters*

Service 20h: Match

TTY Function

Input

AH = 20h
AL = option — 00h – start match; 01h – abort match
BX = Session ID
CX = length of Match String
ES:DI = address of Match String

Output

AL = Return Code — 00h – successful; 01h – invalid session id; 02h – invalid string length

Description

This function attempts to match the Match String with the input data stream. This function returns immediately, applications should use *Service 1F: Query Status* to determine if the match is successful or not. The Match String may not exceed 128 bytes. It only needs to be supplied for option 00 (start match).

see ☞*Access/X.25 User's Guide – Script Commands & Their Syntax*

Note

Only one session at a time may have a match in progress.

Service 18h: Query Cursor

TTY Function

Input

AH = 18h
AL = 00
BX = Session Id

Output

AL = Return Code

00h – successful
01h – invalid session id

DL = column
DH = row

Description

This function returns the current cursor position in an ITI or VT100 session.

Service 1Fh: Query Status

TTY Function

Input

AH = 1Fh
AL = 00
BX = Session ID

Output

AL = Return Code
- 00h – successful
- 01h – invalid session id

CX = Status Word
DL = number of rows
DH = number of columns

Description

This function returns a Status Word that allows applications to monitor the progress of API calls. The Status Word format is:

Status Word

Bit	Meaning	Notes
0	sting match successful	
1	string match in progress	a successful match sets this bit to 0
2	session active	a connection has been established
3	file transfer in progress	
4	binary file transfer/receive in progress	
5	file capture enabled	
6	send buffer is full	cannot send keystoke
7	receive buffer is empty	nothing to read
8-F	reserved	

Service 1Bh: Read Buffer Character

TTY Function

Input

AH = 1Bh
AL = 00
BX = Session ID
DX = Screen Buffer Offset

Output

AL = Return Code

00h – successful
01h – invalid session id
10h – invalid offset

CL = Character

Description

This service returns a single character from the TTY buffer. The character is in ASCII format.

Service 19h: Read Next Character

TTY Function

Input

AH = 19h
AL = 00
BX = Session ID

Output

AL = Return Code

00h – successful
01h – invalid session id
or session not on hold
10h – no data to read

CL = Character

Description

The session must be put on Hold (*Service 04: Background*) before this function can be used. This function intercepts the next character as it arrives from the remote and returns it in register CL. The character is not passed to Access (it will not be placed into the screen buffer or if it is a movement key it will not be interpreted).

Service 1Ch: Read String

TTY Function

Input

AH = 1Ch
AL = 00
BX = Session ID
CX = length of the String
DX = Screen Buffer Offset
ES:DI = address of a buffer to receive the String

Output

AL = return code

00h – successful
01h – invalid session id
10h – invalid offset

Description

This function returns an ASCII String of the indicated length from the TTY Screen Buffer.

Note

If the return string exceeds the end of the Screen Buffer (Screen Buffer Offset + length of string > size of Screen Buffer), random characters will be returned for all positions beyond the edge of the Screen Buffer.

Service 18h: Send Keystroke

TTY Function

Input

AH = 18h
AL = 00
BX = Session ID
CX = KeyID of the keystroke

Output

AL = Return Code

00h – successful
01h – invalid session id
10h – invalid key value or TTY buffer full

Description

This function allows application programs to mimic user keyboard entry for TTY and VT100 sessions. All keys sent by this function will produce the same results as if typed on the keyboard which is dependent on the current state of the PAD parameters. For example, if echo is on, keystrokes will be echoed to the screen.

In addition to regular keys, special keys such as cursor moves and PF1 may be sent

see ☞*Access/X.25 User's Guide – Appendix B. ITI PAD Parameters*

Service 21h: Set PAD Parameters

TTY Function

Input

AH = 21h
AL = read/write flag — 00h – read PAD parameters; 01h – write PAD parameters
BX = Session ID
CX = length of PAD String
ES:DI = address of PAD String

Output

AL = return code — 00h – successful; 01h – invalid session id; 02h – invalid length; 03h – invalid PAD parameter

Description

This function allows you to set PAD parameters for the specified session. The PAD String cannot be longer then 128 bytes and should follow the format specified in the Access User's Guide.

see ☞*Access/X.25 User's Guide – Appendix B. ITI PAD Parameters*

Notes

The PAD String is only needed when the read/write flag is 01.

Executable DOS Files

The section provides a detailed description of each of the Executable DOS files. There are 19 files. They are arranged in alphabetical order.

3270FT: 3270 File Transfer

3270 Command

Summary

3270FT { "*Command String*" | **/STOP** } [**/I**=*Interrupt*]

Return Value

ERRORLEVEL = 00 – successful
01 – no active sessions or invalid session type
04 – unable to enter command in field
05 – unable to open file
06 – syntax error in command string
07 – not allowed to terminate
10 – invalid parameter
11 – Access API not resident

Description

This command allows the user to initiate and terminate a 3270 file transfer in the current foreground session. To initiate the transfer a Command String must be supplied enclosed in quotes. This string should be in the same format as entered when using the Access application program interactively. **see** ☞*Access/QLLC (3270 Support) or Access/SDLC (3270 Support) User's Guide – The F1 File Transfer Key*

The /STOP switch aborts any active file transfer.

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

Note

This command returns as soon as the file transfer has started. Use WAIT 3270FT to determine when the transfer is complete.

ANSWER: Create an ITI Answer Session *ITI Command*

Summary

ANSWER *Name* [/I=*Interrupt*]

Return Value

ERRORLEVEL = 00 – successful
01 – name not found
02 – answer unsuccessful
10 – invalid parameter
11 – Access API not resident

Description

This function creates an ITI answer session. The Name must exist in the Access calling directory. This function returns immediately without waiting for completion of the connection. Use the *WAIT* command to determine when the incoming call has connected.

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

Note

This function is for ITI sessions only.

CALLS: Make a Call

3270, 5250, ITI Command

Summary

CALL *Name* [/I=*Interrupt*]

Return Value

ERRORLEVEL = 00 – successful
01 – name not found
02 – call failed
10 – invalid parameter
11 – Access API not resident

Description

This function makes a 3270, 5250, or ITI call. If Name exists in the Access calling directory, this function will try to make the call using the parameters it finds. If the call is successful then this session becomes the current foreground session.

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

Note

This function only returns once the call has been completed or rejected. Also, if Name is that of an Autolog (.ALG) file, the file will be opened and executed exactly as if it were being run under Access. Depending on the nature of the Autolog file this might lead to screen updates being enabled and messages being written to the display.

see ☞ *Access/X.25 – Autostart & Autolog Features*

COMPARE: String Comparison

3270, 5250, TTY Command

Summary

COMPARE "*String*" *Offset* [/I=*Interrupt*]

Return Value

ERRORLEVEL = 00 – successful
01 – no active sessions or invalid session type
02 – no match
10 – invalid parameter
11 – Access API not resident

Description

This function compares the characters in String with those starting at Offset in the Screen Buffer of the current foreground session.

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

CAPTURE: Capture TTY File

TTY Command

Summary

CAPTURE { *Device Name* | **/OFF** } [*/I=Interrupt*]

Return Value

ERRORLEVEL = 00 – successful
01 – no active sessions or invalid session type
03 – already in capture mode
05 – unable to open target file
10 – invalid parameter
11 – Access API not resident

Description

This function turns TTY file capture on or off. Device Name can be the name of a file or a printer port (LPT1, LPT2).

see ☞ *Access/X.25 – Autostart & Autolog Features*

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

ENDAPI

3270, 5250, TTY Command

Summary

ENDAPI [/I=*Interrupt*]

Return Value

none

Description

This function closes the API interface. Since each of the Executable DOS functions does this automatically, this function should only be used when a function is prematurely aborted.

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

GETSCR: Display Screen Buffer

3270, 5250, TTY Command

Summary

GETSCR *Offset Number Length* [*Session Number*] [/I=*Interrupt*]

Return Value

ERRORLEVEL = 00 – successful
01 – no active sessions or invalid session type
10 – invalid parameter
11 – Access API not resident

Description

This function displays characters from the screen buffer of the selected session. If no session is specified the current foreground session is used. Characters are displayed beginning at location Offset and ending at location Offset + Length. The output of this function is displayed on the screen, however it can be redirected to a file or printer using the DOS redirection commands.

see ☞ *IBM Disk Operating System Reference – Standard Input and Standard Output*

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

Examples

GETSCR 1 4 9	Returns 4 characters starting at position 1 in session number 9.
GETSCR 10 80 > SCR.TXT	Sends 80 characters starting at position 10 in the current foreground session to the file SCR.TXT.
GETSCR 0 1919 lpt1	Sends the entire screen of the current foreground session to the printer.

GETSTAT: Query Access Type

3270, 5250, TTY Command

Summary

GETSTAT [/I=*Interrupt*]

Return Value

ERRORLEVEL = 101– X25
102– QLLC
106– SDLC

Description

This function allows the user to determine the option of the Access software currently running.

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

HANGUP: Hang up a Session

3270, 5250, TTY Command

Summary

HANGUP [/I=*Interrupt*]

Return Value

ERRORLEVEL = 00 – successful
01 – no current session
11 – Access API not resident

Description

This function hangs up the current foreground session. If there are other active sessions, the SWITCH command can be used to bring another session to the foreground.

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

Note

This function does not return until the session has been hung up. If there is a communication problem on the line that hampers the hang up procedure, the function might hold for a long time before completing.

HOTKEY: Hot Key Switch

3270, 5250, TTY Command

Summary

HOTKEY [/I=*Interrupt*]

Return Value

ERRORLEVEL = 00 – successful
11 – Access API not resident

Description

This service causes the Access program to be activated in interactive mode. Access comes up with the current foreground session active and with keyboard input and screen updates enabled.

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

Note

There is no mechanism to prevent the user from Hot Keying out of Access as with the Assembler function (*Service 13h: Hot Key Switch*).

MATCH: Match Input Stream

TTY Command

Summary

MATCH { ["*String*"] [*/Hex Value*] [*/Attribute*] | **/STOP** }
[/I=*Interrupt*]

Return Value

ERRORLEVEL = 00 – successful
01 – no active sessions or invalid session type
03 – match in progress
10 – invalid parameter
11 – Access API not resident

Description

This function tries to match the specified String, Hex Value, Attribute or a combination of all three with the input data stream being received by the current foreground session. The WAIT command should be used to determine if the match is successful or not.

Hex Value is a two-digit hexadecimal number. Attribute can be the characters CR, LF or TAB.

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a number indicating the new setting.

Example

The following is an excerpt from a DOS batch file. It illustrates how to use the MATCH command in combination with the WAIT command.

MATCH "Password:" /TAB	Try to match the character string "Password" followed by a tab.
WAIT MATCH 30	Wait until the match completes or 30 seconds has elapsed.
IF ERRORLEVEL 1 GOTO STPMAT	Jump to STPMAT if the ERRORLEVEL is 1 or greater.
GOTO NEXT	The ERRORLEVEL was 00 indicating success. Jump to the label NEXT which would continue execution of the batch file
:STPMAT MATCH /STOP	Stop looking for a match.
GOTO QUIT	Exit the batch file

PUTSCR: Write Data to Session

3270, 5250, TTY Command

Summary

PUTSCR [*Session Number*] [/ **I**=*Interrupt*]

Return Value

ERRORLEVEL= 00 – successful
01 – no active session
or invalid session id
10 – invalid parameter
11 – Access API not resident
16 – input inhibited
17 – 5250 operator error

Description

This command copies data from STDIN (standard input, usually the keyboard) to the current cursor position in the foreground session. The process stops when end-of-file is encountered (CTRL-Z).

see ☞ *IBM Disk Operating System Reference – Standard Input and Standard Output*

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

Note

STDIN can be redirected to be a file or even another function.

Examples

PUTSCR 1	Accept input from the keyboard and place it into the current foreground session. This process continues until CTRL-Z and RETURN are pressed.
PUTSCR 1 < MESSAGE	Copy the characters in the file MESSAGE to session 1.
GETSCR 0 80 2| PUTSCR	Grab 80 characters from session 2 and send them to the current foreground session.

SCRSAVE: Save Screen to File

3270, 5250 Command

Summary

SCRSAVE *Filename* [*/I=Interrupt*]

Return Value

ERRORLEVEL = 00 – successful
01 – no active sessions or invalid session type
05 – unable to open file
10 – invalid parameter
11 – Access API not resident

Description

This function saves an image of a 3270 or 5250 session screen to a file.

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

SEND: Send TTY File

TTY Command

Summary

SEND { [**/ASCII**] *Filename* | **/STOP** } [**/I**=*Interrupt*]

Return Value

ERRORLEVEL = 00 – successful
01 – no active sessions
or invalid session type
05 – unable to open the file
06 – already sending
10 – invalid parameter
11 – Access API not resident

Description

This function initiates and terminates TTY file transfers in the current foreground session. The /ASCII switch sends the file in ASCII format, otherwise the transfer occurs in binary format.

The /STOP switch aborts the transfer.

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

Note

This command returns as soon as the file transfer has started. Use the WAIT command to determine when the transfer is complete.

SETCURS: Set Cursor Position

3270, 5250 Command

Summary

SETCURS *Column Row* [/I=*Interrupt*]

Return Value

ERRORLEVEL = 00 – successful
01 – no active sessions or invalid session type
02 – invalid input data
10 – invalid parameter
11 – Access API not resident
16 – input inhibited
17 – 5250 operator error

Description

This function sets the cursor position in current foreground session.

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

SWITCH: Switch Session

3270, 5250, TTY Command

Summary

SWITCH [*Name*] [/I=*Interrupt*]

Return Value

ERRORLEVEL = 00 – successful
01 – no sessions
11 – Access API not resident
14 – Name not found

Description

This command brings the session with Name to the foreground. If no name is specified then the next foreground session becomes the session according to growing Session ID.

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

TYPES: Send Keystrokes

3270, 5250, TTY Command

Summary

TYPES ["*String*"] [/Function Key] [/Hex Value] [/I=*Interrupt*]

Return Value

ERRORLEVEL = 00 – successful
01 – no active session or invalid session type
10 – invalid function key
11 – Access API not resident
17 – 5250 operator error
18 – AID key successfully sent

Description

This command can be used to send a String, a Function Key, a Hex Value or any combination of the three to the current foreground 3270, 5250, and TTY session.

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

A String is ASCII characters delimited by quotes.
Hex Value is a one- or two-digit hex number.
Function Key is a value from one of the following tables:

VT100 Function Keys

Key	Description	Key	Description
BKSP	(backspace)	N2	(numeric 2)
DEL	(delete)	N3	(numeric 3)
CURDN	(cursor down)	N4	(numeric 4)
CURLT	(cursor left)	BKSP	(backspace)
CURRT	(cursor right)	N5	(numeric 5)
CURUP	(cursor up)	N6	(numeric 6)
ENTR	(numeric enter)	N7	(numeric 7)
ESC	(escape)	N8	(numeric 8)
LF	(line feed)	N9	(numeric 9)
N-	(numeric -)	PF1	
N,	(numeric ,)	PF2	
N.	(numeric .)	PF3	
N0	(numeric 0)	PF4	
N1	(numeric 1)	RET	(return)

ITI Function Keys

Key	Description	Key	Description
CR	(carriage return)	TAB	(horizontal tab)
LF	(line feed)		

3270 Function Keys

Only one of these AID keys may be used per TYPES command and it must be the last keystroke sent to the session.

Key	Description	Key	Description
PA1		PF10	
PA2		PF11	
PA3		PF12	
ATTN	(attention)	PF13	
SYSREQ	(system request)	PF14	
CURSEL	(cursor select)	PF15	
ENTER		PF16	
CLEAR		PF17	
PF1		PF18	
PF2		PF19	
PF3		PF20	
PF4		PF21	
PF5		PF22	
PF6		PF23	
PF7		PF24	
PF8			
PF9			

The following AID keys may be repeated.

Key	Description	Key	Description
CLEAR		ERSINP	(erase input)
CURDN	(cursor down)	FASTLT	(fast cursor left)
CURLT	(cursor left)	FASTRT	(fast cursor right)
CURRT	(cursor right)	FM	(field mark)
CURUP	(cursor up)	HOME	
DEL	(delete)	NL	
DUP		RESET	
CLEAR		RTAB	(right tab)
ERSEOF	(erase end of file)	TAB	

5250 Function Keys

Only one of these AID keys may be used per TYPES command and it must be the last keystroke sent to the session.

Key	Description	Key	Description
ATTN	(attention)	ROLLUP	(roll up)
BKSP	(backspace)	SYSRQ	(system request)
CLEAR		TEST	
CMD	(command mode)		
COLOR		PF1	
CURDN	(cursor down)	PF2	
CURLT	(cursor left)	PF3	
CURUP	(cursor up)	PF4	
CURRT	(cursor right)	PF5	
DEL	(delete)	PF6	
ERSINP	(erase input)	PF7	
DUP	(duplicate)	PF8	
ENTER	(record advance)	PF9	
FADV	(field advance)	PF10	
FASTLT	(fast cursor left)	PF11	
FASTRT	(fast cursor right)	PF12	
FBKSP	(field backspace)	PF13	
FEXIT	(field exit)	PF14	
FMINUS	(field minus)	PF15	
FPLUS	(field plus)	PF16	
HELP		PF17	
HEX	(hexadecimal)	PF18	
HOME		PF19	
INS	(insert)	PF20	
NL	(new line)	PF21	
PRINT		PF22	
RESET	(errror reset)	PF23	
ROLLDN	(roll down)	PF24	

WAIT: Wait for an Event

3270, 5250, TTY Command

Summary

WAIT { SEND | MATCH | 3270FT | ANSWER | RESPONSE } [Delay]
[/I=*Interrupt*]

Return Value

ERRORLEVEL = 00 – successful
01 – no current or invalid session type
02 – time expired
11 – Access API not resident
16 – Input inhibited
17 – 5250 operator error
18 – AID key sent successfully

Description

This function allows a user to wait until an event completes. Delay controls the length of time (in seconds) this command will wait. If no Delay is specified, the function waits until the event occurs. Delay can be used alone without any of the other options to pause execution of a series of commands.

Each keyword forces this function to wait for an event as follows:

SEND	waits for completion of a TTY file transfer
MATCH	waits for successful match
3270FT	waits for a 3270 file transfer to complete
ANSWER	waits for the current foreground session, which is in waiting state, to receive its connection
RESPONSE	waits for data to be received from a 3270 host and Input Not Inhibited

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

WRSTRING: Write String

3270, 5250 Command

Summary

WRSTRING "*String*" *Offset* [*/I=Interrupt*]

Return Value

ERRORLEVEL= 00 – successful
01 – no active sessions or invalid session type
02 – invalid offset
06 – invalid character in string
10 – invalid parameter
11 – Access API not resident
14 – protected field encountered
16 – input inhibited
17 – 5250 operator error

Description

This function writes a string of characters to the Screen Buffer of the current 3270 or 5250 session. Offset is the Screen Buffer offset where the write will begin. String is an ASCII string delimited by quotes.

The /I switch is tells the API interface that the software interrupt has been altered. It is followed by a hexadecimal number indicating the new setting.

Appendix A: The KeyID Number

The KeyID number allows identification of any function key on the 3270, 5250 and VT-100 keyboards, as well as any key on the PC keyboard. It is a four-digit hexadecimal number.

3270 Keys

Function Key	KeyID	Function Key	KeyID
PA1	0200h	PF17	021Fh
PA2	0201h	PF18	0220h
DUP	0202h	PF19	0221h
FIELD MARK	0203h	PF20	0222h
ATTENTION	0204h	PF21	0223h
SYSTEM REQUEST	0205h	PF22	0224h
CLEAR	0206h	PF23	0225h
ERASE EOF	0207h	PF24	0226h
CURSOR SELECT	0208h	PA3	0227h
ERASE INPUT	0209h	ALT CURS	0228h
RESET	020Ah		
ENTER	020Bh	**Extended Keys**	
NEW LINE	020Ch	EH_DEFAULT	0229h
FAST CURSOR RT	020Dh	EH_BLINK	022Ah
FAST CURSOR LT	020Eh	EH_REVERSE	022Bh
PF1	020Fh	EH_UNDERSORE	022Ch
PF2	0210h	COLOUR_DEFAULT	022Dh
PF3	0211h	BLUE	022Eh
PF4	0212h	RED	023Fh
PF5	0213h	PINK	0230h
PF6	0214h	GREEN	0231h
PF7	0215h	TURQUOISE	0232h
PF8	0216h	YELLOW	0233h
PF9	0217h	WHITE	0234h
PF10	0218h	PS_DEFAULT	0235h
PF11	0219h	PS_A	0236h
PF12	021Ah	PS_B	0237h
PF13	021Bh	PS_C	0238h
PF14	021Ch	PS_D	0239h
PF15	021Dh	PS_E	023Ah
PF16	021Eh	PS_F	023Bh

5250 Non Command Mode Keys

The following are non-command mode function keys. An attempt to enter them while in command mode will result in an error.

Function Key	KeyID	Function Key	KeyID
CURSOR DOWn	0600h	COLOR	061Bh
CURSOR UP	0601h	HEXADECIMAL	061Ch
CURSOR RIGHT	0602h	CLEAR	061Dh
CURSOR LEFT	0603h	TEST	061Eh
FAST CURSOR RT	0604h	PF1	061Fh
FAST CURSOR LT	0605h	PF2	0620h
BACKSPACE	0606h	PF3	0621h
NEWLINE	0607h	PF4	0622h
FIELD ADVANCE	0608h	PF5	0623h
FIELD BACKSPACE	0609h	PF6	0624h
FIELD EXIT	060Ah	PF7	0625h
FIELD PLUS	060Bh	PF8	0626h
FIELD MINUS	060Ch	PF9	0627h
ENTER/RECORD		PF10	0628h
ADVANCE	060Dh	PF11	0629h
ROLL DOWN	060Eh	PF12	062Ah
ROLL UP	060Fh	PF13	062Bh
HOME	0610h	PF14	062Ch
DELETE	0611h	PF15	062Dh
INSERT	0612h	PF16	062Eh
DUPLICATE	0613h	PF17	062Fh
ERASE INPUT	0614h	PF18	0630h
SYSTEM REQUEST	0615h	PF19	0631h
ATTENTION	0616h	PF20	0632h
HELP	0617h	PF21	0633h
PRINT	0618h	PF22	0634h
ERROR RESET	0619h	PF23	0635h
COMMAND MODE	061Ah	PF24	0636h

5250 Command Mode Keys

The following are command mode function keys. An attempt to enter any of them while not in command mode results in an error.

Function Key	KeyID	Function Key	KeyID
RESET COMMAND		PF10	0645h
MODE	0637h	PF11	0646h
COLOR	0638h	PF12	0647h
HEXADECIMAL	0639h	PF13	0648h
CLEAR	063Ah	PF14	0649h
TEST	063Bh	PF15	064Ah
PF1	063Ch	PF16	064Bh
PF2	063Dh	PF17	064Ch
PF3	063Eh	PF18	064Dh
PF4	063Fh	PF19	064Eh
PF5	0640h	PF20	064Fh
PF6	0641h	PF21	0650h
PF7	0642h	PF22	0651h
PF8	0643h	PF23	0652h
PF9	0644h	PF24	0653h
PF10	0645h		

VT100 Keys

The following are command mode function keys. An attempt to enter any of them while not in command mode results in an error.

Function Key	KeyID	Function Key	KeyID
PF1	0200	numeric 3	020E
PF2	0201	numeric 4	020F
PF3	0202	numeric 5	0200
PF4	0203	numeric 6	0201
numeric -	0204	numeric 7	0202
numeric ,	O205	numeric 8	0203
numeric .	0206	numeric 9	0204
numeric ENTER	0207	CURSOR LEFT	0205
BACKSPACE	0208	CURSOR RIGHT	0206
DELETE	0209	CURSOR UP	0207
LINE FEED	020A	CURSOR DOWN	0208
numeric 0	020B	RETURN	0209
numeric 1	020C	ESCAPE	020A
numeric 2	020D		

Appendix B: Modifying Executable DOS Files

The Executable DOS files provide batch file access to Access API functions. Each Executable file is a small program written in the C language that implements one API function. To keep things simple each function opens and closes the API interface when it is run. This gives the user complete freedom when making use of the programs.

Design

The Executable DOS files were written with the Microsoft C Compiler Version 4.0. If you are going to use another compiler to recompile the code, you might have to make some minor alterations to the files so they will compile properly.

The source code for all the Executable DOS files is located in the SOURCE subdirectory on the API diskette(s). They have the file extension .C. In addition to these files, there are two other files that were used to help develop the Executable DOS Files. They are UTILS.C and API.H.

UTILS.C

This file contains the source code for six utility functions that are used repetitively by all the other Executable DOS Files. Compile this file and link it with the other programs when you modify them.

API.H

This header file defines constants used by all the programs. It is "included" at the beginning of each source code file.

Source Code for UTILS.C

The following is commented source code for UTILS.C. Study it before you begin to modify the Executable DOS Files or write your own C functions.

```
/*  utils.c    Copyright (c)  Eicon Technology Corp. 1988
    Description:   This file contains six functions that are used repetitively
                   by the other Executable DOS files. They are:

                        start_api( )
                        end_api( )
                        ret_current_id( )
                        set_fg( sessid )
                        set_interrupt( argc_p, argv_p )
                        hex_value( code )

    Usage:  Compile this file and link the resulting .OBJ file with the other
            API programs.
*/
#include  <dos.h>
#include <string.h>
#include  "api.h"
unsigned char  int_value;  /* current value of the software interrrupt */

/*****************************************************************************/
/* start_api : This function starts the API interface using Service 00h. It
checks to make sure that the current software interrupt is not set to zero
and that Access API is indeed at the other end of the interrupt.
*/
char start_api()
{
    char bufstat[65];
    int res, buf[2];
    struct SREGS sgregs;
    union REGS registers;
    unsigned int destseg, srcoff, bufadr;
/* Check if the interrupt vector for the current software interrupt is not
zero. If it is zero and we issue an interrupt we will crash the system. Note
that at this point we are still not sure that Access API has been loaded.
All we know is that the interrupt exists. It could be some other program. */
    destseg = sgregs.ds;
    segread( &sgregs );
    srcoff = int_value * 4;
    bufadr = buf;
    movedata( 0, srcoff, destseg, bufadr, 4);
    if ( buf[0] == 0 && buf[1] == 0 )
      return( 1 );
```

continued next page

"utils.c" continued

```
/*  Start the API Interface using Service 00H. This lets us know if Access
API is loaded. */
    registers.x.ax = START;
    res = int86x ( int_value, &registers, &registers, &sgregs );
    if ( registers.h.al != SUCCESS )
      return (1);
/*  Double check to confirm we are indeed talking to Access through the in-
terrupt. To do this we check the session status of an invalid session and
verify the result. */
    registers.x.ax = SESSION_STAT;
    registers.x.bx = -1;    /* Invalid session id */
    registers.x.di = bufstat;
    sgregs.es = sgregs.ds;
    res = int86x ( int_value, &registers, &registers, &sgregs );
    if ( registers.h.al == INVAL_ID )
      return(0);
    else
      return( 1 );
}
/*****************************************************************************/
/* set_interrupt : This function strips the /interrupt flag and correspond-
ing hex value from the command line. If the hex value is legal, then the
global variable int_value is set. This ensures that all API function calls
will use the proper software interrupt.
*/
set_interrupt (argc_p, argv_p)
int *argc_p;
char **argv_p[];
{
    char **parm;
    int  i;
    int_value = INT_NB;
    if ( *argc_p <2 )
      return;
    parm = *argv_p;
    if ( strlen(  *( parm+1 ) )  <4 )
      return;
    if ( strncmp( *( parm+1 ),"/I=",3 ) )
      if (strncmp ( *(parm+1),"/i=",3 ) )
        return;
    i = hex_value( *( parm+1 )+2  );
    if ( i == NOT_HEX )
      return;
    if ( i<0x40 )
      return;
    ( *argc_p )--;
    ( *argv_p )++;
    int_value = i;
    return;
}
```

continued next page

"utils.c" continued

```
/*****************************************************************************/
/* ret_current_id : Returns the Session ID of the current foreground session.
*/
int ret_current_id()
{
    char rc, bufstat[65];
    int res, i;
    union REGS registers;
    struct SREGS sgregs;

    segread( &sgregs );
    sgregs.es  = sgregs.ds;

/* Use Service 11: Query Status to find the current foreground session.
Register CL will be 1 if session is in foreground. Since Access supports
up to nine concurrent sessions we must loop, checking each one in succession */

    for ( i=1; i; i++ )
    {
      registers.x.ax = SESSION_STAT;
      registers.x.di = bufstat;
      registers.x.bx = i;
      res = int86x ( int_value, &registers, &registers, &sgregs );

      if ( (registers.h.al == SUCCESS) && (registers.h.cl == 1) )  /* found it */
        return( i );
    }
    return( 0 );/*  no foreground session */
}
/*****************************************************************************/
/* end_api  :  This function closes the API Interface using Service 14H.
*/
char  end_api()
{
    int res;
    union REGS registers;

    registers.x.ax = END;
    res = int86( int_value, &registers, &registers );
}
/*****************************************************************************/
/* set_fg : This function places the session with sessid into the
foreground.*/
int set_fg( sessid )
int sessid;
{
    int    res;
    union  REGS registers;

    registers.x.ax = CSET_FG;
    registers.x.bx = sessid;
    res = int86( int_value, &registers, &registers );
```

continued next page

"utils.c" continued

```
/*****************************************************************************/
/* hex_value : This function returns the integer value of a two-digit
hexadecimal number (code). The format of code is a slash followed by
two ascii characters.
*/

int hex_value( code )
char *code;

{
    int i, value = 0;
    char c;

    code++;      /* advance to ignore the slash */
    for ( i=0; i; i++ )
    {
      c = code[i];
      if ( c == 0 )
        return( value );
      value = value < 4;
      if ( (c>='a') && (c <='f') )
        value += ( c - 'a' ) + 0x0a;
      else
      if ( (c>='A') && (c<='F') )
        value += ( c - 'A' ) + 0x0a;
      else
      if ( (c>='0') && (c<='9') )
        value += ( c - '0' );
      else
      return( NOT_HEX );  /* not  a hex value */
    }
    return( value );
}
```

end of "utils.c"

Appendix C: Sample Programs

Assembler Interface

The following is the source code for the ANSWER Executable DOS file.

```
/*  answer.c    Copyright (c)  Eicon Technology Corp. 1988
Description:   This program initiates a 3270, 5250 or ITI call. The call
name must exist in the Access calling directory. This program returns
without waiting for the call to complete. The WAIT program should be
used to determine if the call has connected.

Usage:  ANSWER call name.
*/
#include  <dos.h>
#include <stdio.h>
#include "api.h"
extern unsigned char int_value;
main( argc, argv )
int argc;
char *argv[];
{
    char rc, start_api(), end_api();
    int res;
    union REGS  registers;
    struct SREGS sgregs;
    set_interrupt (&argc, &argv);
    if ( argc != 2 )
    {
      puts("*** Invalid number of parameters");
      exit( INVAL_PARM );
    }
    rc = start_api();
    if ( rc != SUCCESS )
    {
      puts("*** API system not loaded");
      exit( API_NOT_RES );
    }
    registers.x.cx = strlen( argv[1] );
    registers.x.di = argv[1];
    registers.x.ax = CANSWER;
    segread( &sgregs );
    sgregs.es = sgregs.ds;
    res = int86x ( int_value, &registers, &registers, &sgregs );
```

continued next page

```
if ( registers.h.al != SUCCESS )
{
  rc = end_api();
  if ( registers.h.al == 0x01 )
    puts("*** Calling directory name not found");
  if ( registers.h.al == 0x02 )
    puts("*** Answer unsuccessfull")
  exit( registers.h.al );
}
rc = set_fg( registers.x.bx );  /* put the session into the foreground */
rc = end_api();
```

end of "answer.c"

Batch Files

The most effective way to make use of the Executable DOS Files is in batch files. Through the use of the ERRORLEVEL variable it is possible to construct some fairly sophisticated batch files to automate repetitive tasks.

When you are testing ERRORLEVEL values make sure that you start with the highest value and work your way down. The IF expression will be true if ERRORLEVEL is equal to or less than the test value.

Sample Batch File #1

Batch File Entry	Notes
getstat	Determine Access program type
if errorlevel = 106 goto SDLC	Test for match in decending order
if errorlevel = 102 goto QLLC	
if errorlevel = 101 goto X25	
if errorlevel = 11 goto quit	API not resident so quit.
:X25	
echo Access/X.25	
goto quit	
:QLLC	
echo Access/QLLC	
goto quit	
:SDLC	
echo Access/SDLC	
:quit	

The following is an example of a batch file that could be used by a branch office to exchange daily reports with its head office.

Sample Batch File # 2

Batch File Entry	Notes
echo off	
answer HEADOFFICE	Setup to receive a call from the Head Office
if errorlevel = 1 goto end	Any value >= 1 indicates an error condition, so exit
wait ANSWER 30	Wait 30 seconds maximum for call to connect
if errorlevel = 1 goto end	Any value >= 1 indicates an error condition, so exit
match "User name?"	Look for the prompt
if errorlevel = 1 goto stop	
wait MATCH 30	Wait 30 seconds for the prompt
if errorlevel = 1 goto stop	
type "Office25" /CR	Respond to the prompt with a user name plus a carriage return
if errorlevel = 1 goto stop	
capture "c:\report"	Enable capture of data to a file
if errorlevel 1 goto stop	
match "Report Sent'"	Look for confirmation
if errorlevel = 1 goto stop	
wait MATCH 120	Wait 120 seconds maximum for file transfer to complete and confirmation to be displayed
if errorlevel = 1 goto stop	
send "c:\news.txt"	Send a file to Head Office
if errorlevel = 1 goto stop	
wait SEND	Wait for the send to be completed
stop:	
hangup	An error occured once the connection was established so hang it up before quitting.
end:	Return to DOS

Appendix D: Reference Books

Eicon Technology

The following manuals are related to the operation and use of different versions of the Access application program. The manuals are available from Eicon Technology or any of its representatives.

- Access/X.25 User's Guide (200-100-3)
- Access/QLLC 3270 Support User's Guide (200-143-1)
- Access/QLLC 5250 Support User's Guide (200-126-1)
- Access/SDLC 3270 Support User's Guide (200-108-2)
- Access/SDLC 5250 Support User's Guide (200-129-1)

IBM

The following is a list of reference books related to IBM software and hardware products:

5250 Communications

- IBM 3251 Display Station Models 2 and 12 Operators Guide (GA21-9323)
- IBM 5251 Display Station Models 1 and 11 Operator's Guide (GA21-9248)
- IBM 5291 Models 1 and 2 Display Station User's Guide (GA21-9409)
- IBM 5292 Color Display Station Operator's Guide (GA21-9416)
- System Network Architecture Concepts and Products (GC30-3072-2)
- IBM 5250 Information Display System Functions Reference Manual (SA21-9247)
- PC Support/36 User's Guide (SC21-9088-2)
- PC Support/38 User's Guide (SC21-9089-1)

3270 Communications

- IBM 3270 Information Display System: 3274 Control Unit Operator's Guide (GA23-0023)
- IBM 3270 Information Display System 3278 Display Station Operator's Guide (GA27-2890-3)
- IBM PC 3270 Emulation Program User's Guide (59X9951)
- IBM PC 3270 Emulation Program Application Programming Interface and Host Reference (59X9971)
- IBM 3287 Printer Operator's Guide (GA27-3147)

X.25

The X.25 Standards referred to in this manual are defined in various sections of Volume 8 of the CCITT Red Book, Malaga-Torremolinos 1984:

- Volume VIII, Fascicle VIII.2 contains Recommendation X.3
- Volume VIII, Fascicle VIII.3 contains Recommendations X.25, X.28, X.29
- Volume VIII, Fascicle VIII.4 contains Recommendation X.121

All these publications are available from:

United Nations Bookstore
Room GA 32B
New York, NY 10017

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

LIMITED WARRANTY

Eicon Technology warrants the media on which the program is furnished to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of delivery to you as evidenced by a copy of your receipt.

THE PROGRAM IS PROVIDED AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK TO THE QUALITY AND PERFORMANCE OF THE PROGRAMS IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU (AND NOT EICON TECHNOLOGY) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING REPAIR OR CORRECTION.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM ONE JURISDICTION TO ANOTHER.

Eicon Technology does not warrant that the functions contained in the program will meet your requirements or that the operation of the program will be uninterrupted or error free. However, Eicon Technology warrants the media on which the program is furnished, to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of delivery to you as evidenced by a copy of your receipt.

EICON TECHNOLOGY SOFTWARE LICENSE AGREEMENT

YOU SHOULD CAREFULLY READ THE FOLLOWING TERMS AND CONDITIONS BEFORE OPENING THE DISKETTE PACKAGE. OPENING THE DISKETTE PACKAGE INDICATES YOUR ACCEPTANCE OF THESE TERMS AND CONDITIONS. IF YOU DO NOT AGREE WITH THEM, YOU SHOULD PROMPTLY RETURN THE PACKAGE UNOPENED AND YOUR MONEY WILL BE REFUNDED.

Title to the media on which program is recorded and to documentation in support thereof is transferred to you, but the title to the program is retained by Eicon Technology Corporation. You assume responsibility for the selection of the program to achieve your intended results, and for the installation, use, and results obtained from the program.

License

Under the terms and conditions of this License Agreement you may:

- use the program on a single machine;
- copy the program into any machine readable or printed form for backup or modification purposes in support of your use of the program on the single machine; (certain programs however may include mechanisms to limit or inhibit copying; they are marked copy protected;) copying of documentation and other printed material is prohibited;
- modify the program and/or merge it into another program for your use on the single machine; (any portion of this program merged into another program will continue to be subject to the terms and conditions of this Agreement;)
- transfer the program and license to another party if the other party agrees to accept the terms and conditions of this Agreement. If you transfer the program you must at the same time either transfer all copies whether in printed or machine readable form to the same party or destroy any copies not transferred; this includes all modifications and portions of the program contained in or merged into other programs.

You must reproduce and include the copyright notice on any copy, modification or portion merged into another program.

You may not use, copy, modify, or transfer the program, or any copy, modification or merged portion, in whole or in part, except as expressly provided for in this License Agreement.

If you transfer possession of any copy, modification or merged portion of the program to another party, your license is automatically terminated.

Term

The license is effective until terminated. You may terminate it at any time by destroying the program together with all copies, modifications and merged portions in any form. It will also terminate upon conditions set forth elsewhere in this Agreement or if you fail to comply with any term or condition of this Agreement. You agree upon such termination to destroy the program together with all copies, modifications and merged portions in any form.

Limitations of Remedies

Eicon Technology's entire liability and your exclusive remedy shall be:

- with respect to defective media during the warranty period Eicon Technology will replace media not meeting Eicon Technology's Limited Warranty if returned to Eicon Technology or its authorized representative with a copy of your receipt, or
- if Eicon Technology or its representative is unable to deliver replacement media free of defects in materials and workmanship, you may terminate the Agreement by returning the program and your money will be refunded.

IN NO EVENT WILL EICON TECHNOLOGY BE LIABLE FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE SUCH PROGRAM EVEN IF EICON TECHNOLOGY OR ITS AUTHORIZED REPRESENTATIVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR ANY CLAIM BY ANY OTHER PARTY.

SOME JURISDICTIONS DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

General

You may not sublicense, assign or transfer the license or the program except as expressly provided in this Agreement. Any attempt otherwise to sublicense, assign or transfer any of the rights, duties or obligations hereunder is void.

This Agreement is governed by the laws of the Province of Quebec, Canada.

The parties agree that this Agreement be written in English. Les partis consentent à ce que cette entente soit rédigée en anglais.

Should you have any questions concerning this Agreement, you may contact Eicon Technology in writing at 2196 32nd Avenue, Montreal, Canada H8T 3H7.

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. YOU FURTHER AGREE THAT IT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN US WHICH SUPERSEDES ANY PROPOSAL OR PRIOR AGREEMENT, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN US RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

Product Comment Form

Access API Programmer's Reference (200-124-2)

We would appreciate your comments regarding any problems encountered while using this or any other Eicon Technology product. Please use this form to let us know about your concerns.

Your Comments are Welcome

Submitter

Name: _____

Title: _____

Company: _____

Address: _____

_____

_____

Comments

_____

_____

_____

_____

_____

_____

_____

Reach Us At

Eicon Technology can be reached by phone, telex, fax and mail at:

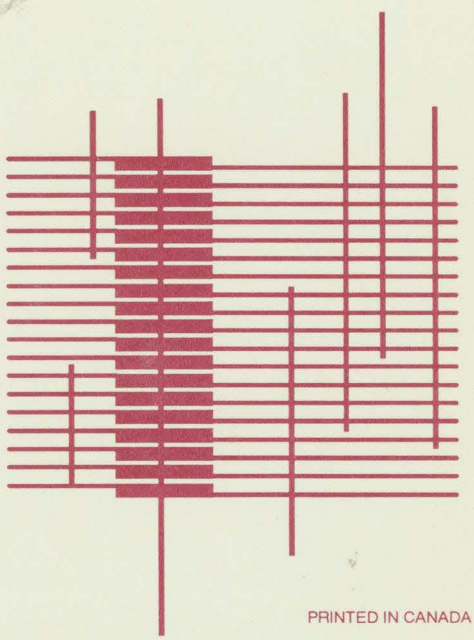
Eicon Technology
2196 - 32nd Avenue (Lachine)
Montreal, Quebec
Canada H8T 3H7

Phone: 514-631-2592
Fax: 514-631-3092
Telex: 05-25134-MTL

This manual was composed with Ventura Publisher® and printed on the Eicon Technology Laser Printer Adapter at 400dpi using EiconScript™.

Eicon Technology Corporation

2196 - 32nd Avenue (Lachine)
Montreal, Quebec
Canada H8T 3H7



PRINTED IN CANADA