Enterprise Access Library

Language Reference

Purpose

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

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

Introduction

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

Attachmate Enterprise Access Library API (EAL) is one such API, developed as a multi-tasking, easier-to-program alternative to emulator APIs available in the early 1990s: EHLLAPI and WinHLLAPI. While those APIs, employed successfully throughout business and industry for a wide range of automation tasks, were *de facto* standards for this class of software, Attachmate’s EALAPI also enjoyed considerable success as an aid to customer automation efforts.

Accessing Attachmate 32-bit EAL API

In brief, an application accesses this interface by ensuring Attachmate software, including dynamic load library ATMAPI32.DLL, is in the system search path, so it will be found and loaded when referenced.

Language support files used by the EAL (.h, .lib, .bas, and so on) are no longer being distributed by Attachmate, so the information here is really only useful for those who are supporting existing EAL applications. Header and lib files for EHLLAPI, WinHLLAPI, and Attachmate HLLAPI are distributed with EXTRA!.

Using window handles

To connect to a session, you must associate that session with a window handle (referred to as the *hWnd* parameter in EAL syntax). Each session you connect to requires a unique window handle and all subsequent function calls require the same *hWnd*.

Refer to the documentation for your development environment for information about retrieving a window handle.

> **Caution** While the originating window (the one whose handle you reference) need not be visible, it must remain loaded because you must pass the same window handle each time you call EAL.

Using the EAL functions

The ATMRegisterClient is always the first function that *must* be called. Each EAL function returns an integer value. Most development environments require you to assign the result of each function to an integer variable, as in this example:

```
ReturnCode = ATMConnectSession(Handle, "A")
```

Your program must check the returned value for possible error conditions. Most functions require a string as one of the parameters. If the function does not return a value in the string parameter, you can pass either a string variable or a literal string (characters enclosed in quotation marks) in the function. For example, you can call the ATMSendString function by entering one of the following:

```
duedate = "July 5, 1997"
ReturnCode = ATMSendString(Handle, 5, 10, duedate)
```

The string variable `duedate` is provided as the *string* parameter, or:

```
ReturnCode = ATMSendString(Handle,5,10,"July 5, 1997")
```

The string literal `July 5, 1997` is provided as the *string* parameter.

For EAL functions that return a string value, you *must* use a string variable. (This requirement applies to most of the functions that contain the word “Get” in the function name.) These functions usually require that you pass the length of the string as well. Use your language’s Length function to determine the length. For example, ATMGetString should be called as follows:

```
ReturnCode = ATMGetString(Handle,Row,Col, Config, Len(Config))
```

ATMUnregisterClient is the last function that *must* be called.

Overview of Functions

Function	Description
ATMAddWait xxx	Adds an event of type xxx to an event table. Available AddWait types include the following: ForCursor ForCursorMove ForHostDisconnect ForKey ForString ForStringNotAt HostQuiet
ATMClearEventTable	Deletes all the events from an event table.
ATMConnectSession	Specifies the host session with which you want to interact.
ATMDeleteEvent	Deletes an event from an event table.
ATMDisconnectSession	Releases the previously connected presentation space.
ATMGetATMAPIVersion	Gets the current version of the ATMAPI dynamic link library.
ATMGetConnectionStatus	Returns host connection information.
ATMGetCursorLocation	Returns the current row and column position of the cursor in the connected host session.
ATMGetEmulatorPath	Returns the path of the currently installed host access software files.
ATMGetEmulatorVersion	Returns the version of the currently installed host access software executable.
ATMGetError	Returns the name of the most recent function call that resulted in an error. Also returns a description of the error.
ATMGetFieldInfo	(3270,5250) Returns information about the specified field.
ATMGetFieldLength	Searches through the currently connected host session for a specified field and returns the field length.
ATMGetFieldPosition	Searches through the currently connected host session for the beginning position of a specified field, and returns the field position.
ATMGetKeystroke	Allows your application to receive the keystrokes for the sessions specified with the function called StartKeystrokeIntercept.
ATMGetLayoutName	Returns the name of the currently loaded host access software layout file.
ATMGetParameter	Returns the current setting of a ATMAPI system parameter.

ATMGetSessionHandle	Returns the window handle of the requested host access software session.
ATMGetSessions	Returns session name information about the available host sessions in the present host access software configuration.
ATMGetSessionSize	Determines the size of the target host session.
ATMGetSessionStatus	Determines whether or not the target session meets certain status criteria.
ATMGetString	Captures a string from a host screen at the specified row and column position.
ATMGetStringFromField	Copies all of the characters from a specified field in the currently connected host session.
ATMHoldHost	(VT only) Allows the user to prevent host data from being received until an ATMResumeHost call is made. This allows processing of received data without disruption from a host application.
ATMListSessions	Generates a list box containing host access software sessions of the type specified in the current configuration and returns the user's session choice.
ATMLockKeyboard	Inhibits user keyboard input in the host session.
ATMOpenLayout	Opens the named host access software layout file.
ATMPause	Pauses the specified number of half seconds.
ATMReceiveFile	Transfers a file from the host to the PC.
ATMRegisterClient	Associates an Attachmate host access product with the client window handle. The connection types are as follows: 1 = EXTRA! (EPC in 32-bit; EXTRA! in 16-bit) 2 = RALLY!™ 3 = KEA! (KEA! 95 in 32-bit; KEA!-VT for 32-bit) 4 = Irma™ 5 = INFOConnect™ 6 = HP (KEA! for HP) This function must be the **first** EAL function call within your application.
ATMResetSystem	Reinitializes the EAL system parameters to their default state and disconnects any previously connected sessions. Clears all event tables created for the hWnd, connected or not.
ATMResumeHost	(VT only.) Allows the user to resume host data reception after a call to ATMHoldHost.
ATMRowColumn	Converts a presentation space position to a row or column position depending on the selection in the option parameter.

ATMSearchField	Searches through the specified protected or unprotected field in the currently connected host session for a specified string.
ATMSearchSession	Searches the presentation space of the currently connected session for the occurrence of a string.
ATMSendAndWait	Sends a keystroke and waits for the specified string to appear in the host presentation space.
ATMSendFile	Transfers a file from the PC to the indicated host session.
ATMSendKey	Sends a host keystroke to the target session.
ATMSendString	Pastes a string of ASCII characters into the presentation space of the connected session.
ATMSendStringToField	Copies a string of characters to a specific unprotected field in the currently connected host session.
ATMSessionOff	Enables a program to power off a specified host session.
ATMSessionOn	Enables a program to power on a specified host session.
ATMSetCursorLocation	Sets the cursor to the location specified in the connected presentation space.
ATMSetParameter	Sets a library system parameter to a new value.
ATMShowLastError	Displays the most ATMAPI32.DLL error in a message box.
ATMStartKeystrokeIntercept	Allows an HLLWin to filter any keystrokes entered by a terminal operator.
ATMStartSession	Enables a program to start a specified configured session.
ATMStopKeystrokeIntercept	Ends the HLLWin's ability to intercept keystrokes for the specified session.
ATMStopSession	Enables a program to stop a specified configured session.
ATMUnlockKeyboard	Unlocks a keyboard previously locked by the ATMLockKeyboard function.
ATMUnregisterClient	Removes the client window handle from the registration table, freeing the supporting HLLAPI DLL and cleaning up associated resources.
ATMWait	Waits for the XCLOCK or XSYSTEM to be cleared from the OIA of the connected session.
ATMWaitForCursor	Waits for the cursor to move to a certain location.
ATMWaitForCursorMove	Waits until the cursor has moved from the current location.

ATMWaitForEvent	Waits for one of the events that has been added to the table to occur and returns a code indicating which event was completed.
ATMWaitForHostConnect	Waits until a host connection is established.
ATMWaitForHostDisconnect	Waits until a host connection is broken.
ATMWaitForKey	Waits until the specified keystroke has been entered by the user (not through the API).
ATMWaitForString	Waits until a specified string appears in the host session.
ATMWaitHostQuiet	Waits for the host to finish processing.

Alphabetic Listing of EAL Functions

ATMAddWait

The ATMAddWait function adds a Wait event to the specified event table. It remains in the table until an ATMDeleteEvent, ATMResetSystem, ATMClearEventTable, or ATMUnregisterClient is issued. The event ID must be unique in this table for this hWnd.

The Wait event is triggered when XCLOCK has been cleared (3270), Input Inhibited is not displayed (5250) or the host has been idle for the minimum time (1/10th of a second for async).

Prerequisites

ATMConnectSession must be called before using this function.

Syntax

```
int rc = ATMAddWait ( hWnd,table,event)
```

Call parameters

Each parameter of the ATMAddWait function is described below.

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
table	This integer indicates the table of events to be waited for.
event	The identifying integer that is returned by ATMWaitForEvent if this event is triggered first.

Return parameters

Result code	
1 or ATM_SUCCESS	The event was added.
-1 or ATM_NOTCONNECTED	The session is not connected.
-9 or ATM_SYSTEMERROR	A system error occurred.
-301 or ATM_EVENTALLREADYSET	The event ID supplied is already in use by this application for this table ID.
-302 or ATM_EVENTMAXEXCEEDED	The maximum number of events allowed for a table has already been added to this table.
-303 or ATM_TABLEMAXEXCEEDED	The maximum number of tables for a hWnd has already been used.

ATMAddWaitForCursor

The ATMAddWaitForCursor function adds a WaitForCursor event to the specified event table. This event remains in the table until one of the following is issued: ATMDeleteEvent, ATMResetSystem, ATMClearEventTable, or ATMUnregisterClient. The event ID must be unique in this table for this hWnd.

The WaitForCursor event is triggered when the cursor moves to the specified row and column or if the cursor is already at the specified row and column when the WaitForEvent function is called.

Prerequisites

ATMConnectSession must be called before using this function.

Syntax

```
rc = ATMAddWaitForCursor ( hWnd,table,event,row,column)
```

Call parameters

Each parameter of the ATMAddWaitForCursor function is described below.

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
table	This integer indicates the table of events to be waited for.
event	The identifying integer that is returned by ATMWaitForEvent if this event is triggered first.
row	The integer row the cursor is to move to. If the value for row is zero, ATMWaitForEvent will return when the cursor is in the selected column in any row. Only one row or column can be zero.
column	The integer column the cursor is to move to. If the value for column is zero, ATMWaitForEvent will return when the cursor is in the selected row in any column. Only one row or column can be zero.

Return parameters

Result code	
1 or ATM_SUCCESS	The event was added.
-1 or ATM_NOTCONNECTED	The session is not connected.
-2 or ATM_INVALIDPARAMETER	Both row and column are zero.
-9 or ATM_SYSTEMERROR	A system error occurred.
-301 or ATM_EVENTALLREADYSET	The event ID supplied is already in use by this application for this table ID.
-302 or ATM_EVENTMAXEXCEEDED	The maximum number of events allowed for a table has already been added to this table.
-303 or ATM_TABLEMAXEXCEEDED	The maximum number of tables for a hWnd has already been used.

ATMAddWaitForCursorMove

The ATMAddWaitForCursorMove function adds a WaitForCursorMove event to the specified event table. The event remains in the table until an ATMDeleteEvent, ATMResetSystem, ATMClearEventTable, or ATMUnregisterClient is issued. The event ID must be unique in this table for this hWnd.

The WaitForCursorMove event is triggered when the cursor moves.

Prerequisites

ATMConnectSession must be called before using this function.

Syntax

```
rc = ATMAddWaitForCursorMove ( hWnd,table,event)
```

Call parameters

Each parameter of the ATMAddWaitForCursorMove function is described below.

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
table	This integer indicates the table of events to be waited for.
event	The identifying integer that is returned by ATMWaitForEvent if this is the event that is triggered first.

Return parameters

Result code	
1 or ATM_SUCCESS	The event was added.
-1 or ATM_NOTCONNECTED	The session is not connected.
-9 or ATM_SYSTEMERROR	A system error occurred.
-301 or ATM_EVENTALLREADYSET	The event ID supplied is already in use by this application for this table ID.
-302 or ATM_EVENTMAXEXCEEDED	The maximum number of events allowed for a table has already been added to this table.
-303 or ATM_TABLEMAXEXCEEDED	The maximum number of tables for a hWnd has already been used.

ATMAddWaitForHostDisconnect

The ATMAddWaitForHostDisconnect function adds the WaitForHostDisconnect event to the specified event table. The event remains in the table until an ATMDeleteEvent, ATMResetSystem, ATMClearEventTable, or ATMUnregisterClient is issued. The event ID must be unique in this table for this hWnd.

The WaitForHostDisconnect event is triggered during processing of an ATMWaitForEvent when the host disconnects. It is triggered immediately if the host is already disconnected. Note that this is a disconnect from the host by the host access software. It has nothing to do with the HLLAPI-type disconnect of the application from the host access software.

Prerequisites

ATMConnectSession must be called before using this function.

Syntax

```
rc = ATMAddWaitForHostDisconnect ( hWnd,table,event)
```

Call parameters

Each call parameter f the ATMWaitForHostDisconnect function is described below.

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
table	This integer indicates the table of events to be waited for.
event	The identifying integer that is returned by ATMWaitForEvent if this is the event that is triggered first

Return parameters

Result code	
1 or ATM_SUCCESS	The event was added.
-1 or ATM_NOTCONNECTED	The session is not connected.
-9 or ATM_SYSTEMERROR	A system error occurred.
-301 or ATM_EVENTALLREADYSET	The event ID supplied is already in use by this application for this table ID.
-302 or ATM_EVENTMAXEXCEEDED	The maximum number of events allowed for a table has already been added to this table.
-303 or ATM_TABLEMAXEXCEEDED	The maximum number of tables for a hWnd has already been used.

ATMAddWaitForKey

The ATMAddWaitForKey function adds a WaitForKey event to the specified event table. This event remains in the table until an ATMDeleteEvent, ATMResetSystem, ATMClearEventTable, or ATMUnregisterClient is issued. The event ID must be unique in this table for this hWnd.

The WaitForKey event is triggered when the specified key is pressed on the keyboard.

Prerequisites

ATMConnectSession must be called before using this function.

Syntax

```
rc = ATMAddWaitForKey ( hWnd,table,event,key)
```

Call parameters

Each parameter of the ATMAddWaitForKey function is described below

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
table	This integer indicates the table of events to be waited for.
event	The identifying integer that is returned by ATMWaitForEvent if this is the event that is triggered first.
key	This string contains the desired keystroke. It can be specified as the ASCII code for the keystroke or a code from the sendkey table (for example, @N for new line).

Return parameters

Result code	
1 or ATM_SUCCESS	The event was added.
-1 or ATM_NOTCONNECTED	The session is not connected.
-9 or ATM_SYSTEMERROR	A system error occurred.
-301 or ATM_EVENTALLREADYSET	The event ID supplied is already in use by this application for this table ID.
-302 or ATM_EVENTMAXEXCEEDED	The maximum number of events allowed for a table has already been added to this table.
-303 or ATM_TABLEMAXEXCEEDED	The maximum number of tables for a hWnd has already been used.

ATMAddWaitForString

The ATMAddWaitForString function adds a WaitForString event to the specified event table. This event remains in the table until an ATMDeleteEvent, ATMResetSystem, ATMClearEventTable, or ATMUnregisterClient is issued. The event ID must be unique in this table for this hWnd.

The WaitForString event is triggered when the specified string appears in the given location.

Prerequisites

ATMConnectSession must be called before using this function.

Syntax

```
rc = ATMAddWaitForString (hWnd,table,event,row,column,waitString)
```

Call parameters

Each parameter of the ATMAddWaitForString function is described below

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
table	This integer indicates the table of events to be waited for.
event	The identifying integer that is returned by ATMWaitForEvent if this is the event that is triggered first.
row	The integer row position in the connected host session where waitString will start. If row is set to zero, all rows will be searched.
column	The integer column position in the connected host session where waitString will start. If column is set to zero, all columns will be searched.
waitString	This string contains the text you want to wait for.

Return parameters

Result code	
1 or ATM_SUCCESS	The event was added.
-1 or ATM_NOTCONNECTED	The session is not connected.
-9 or ATM_SYSTEMERROR	A system error occurred.
-301 or ATM_EVENTALLREADYSET	The event ID supplied is already in use by this application for this table ID.
-302 or ATM_EVENTMAXEXCEEDED	The maximum number of events allowed for a table has already been added to this table.
-303 or ATM_TABLEMAXEXCEEDED	The maximum number of tables for a hWnd has already been used.
-306 or ATM_INVALIDROW	The row value is greater than the host session can support, or not an integer greater than zero when column is zero.
-307 or ATM_INVALIDCOLUM	The column value is greater than the host session can support, or not an integer greater than zero when row is zero.
-308 or ATM_STRINGTOOLONG	The waitString is longer than the maximum allowable value.

Remarks

The last three call parameters are treated exactly like the row, column, and waitString parameters of ATMWaitForString.

The row, column, and string parameters define an *area* of the screen. This area can extend from a given non-zero row, non-zero column, and continue up through the presentation space for the length of the supplied string (a “stream select” in async-terminal terms).

If the value for *row* is zero, a “block” area of screen is defined—one which includes all rows, but is bounded on the left and right by the supplied column and the supplied column plus the string length. If *column* is zero, the area consists of one complete row.

ATMAddWaitForStringNotAt

The ATMAddWaitForStringNotAt function adds a WaitForStringNotAt event to the specified event table. This event remains in the table until an ATMDeleteEvent, ATMResetSystem, ATMClearEventTable, or ATMUnregisterClient is issued. The event ID must be unique in this table for this hWnd.

The WaitForStringNotAt event is triggered when the given location no longer contains the specified string.

Prerequisites

ATMConnectSession must be called before using this function.

Syntax

```
rc = ATMAddWaitForStringNotAt (hWnd,table,event,row,
column,waitString)
```

Call parameters

Each call parameter of the ATMAddWaitForStringNotAt function. is described below.

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
table	This integer indicates the table of events to be waited for.
event	The identifying integer that is returned by ATMWaitForEvent if this is the event that is triggered first.
row	This integer defines the row position in the connected host session where waitString will start.
column	This integer defines the column position in the connected host session where waitString will start.
waitString	This string contains the text you want to wait for.

Return parameters

Result code	
1 or ATM_SUCCESS	The event was added.
-1 or ATM_NOTCONNECTED	The session is not connected.
-9 or ATM_SYSTEMERROR	A system error occurred.
-301 or ATM_EVENTALLREADYSET	The event ID supplied is already in use by this application for this table ID.
-302 or ATM_EVENTMAXEXCEEDED	The maximum number of events allowed for a table has already been added to this table.
-303 or ATM_TABLEMAXEXCEEDED	The maximum number of tables for a hWnd has already been used.
-306 or ATM_INVALIDROW	The row value is greater than the host session can support, or not an integer greater than zero when column is zero.
-307 or ATM_INVALIDCOLUM	The column value is greater than the host session can support, or not an integer greater than zero when row is zero.
-308 or ATM_STRINGTOOLONG	The waitString is longer than the maximum allowable value.

Remarks

The last three call parameters are treated exactly like the row, column, and waitString parameters of ATMWaitForString.

The row, column, and string parameters define an *area* of the screen. This area can extend from a given non-zero row, non-zero column, and continue up through the presentation space for the length of the supplied string (a “stream select” in async-terminal terms).

If the value for *row* is zero, a “block” area of screen is defined—one which includes all rows, but is bounded on the left and right by the supplied column and the supplied column plus the string length. If *column* is zero, the area consists of one complete row.

ATMAddWaitHostQuiet

The ATMAddWaitHostQuiet function adds a WaitHostQuiet event to the specified event table. The event ID must be unique in this table for this hWnd. The event remains in the table until an ATMDeleteEvent, ATMResetSystem, ATMClearEventTable, or ATMUnregisterClient is issued.

The WaitHostQuiet event is triggered when the host has been idle for the specified SettleTime.

Prerequisites

ATMConnectSession must be called before using this function.

Syntax

```
rc = ATMAddWaitHostQuiet ( hWnd,table,event,SettleTime)
```

Call parameters

Each call parameter of the ATMAddWaitHostQuiet function is described below.

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
table	This integer indicates the table of events to be waited for.
event	The identifying integer that is returned by ATMWaitForEvent if this is the event that is triggered first.
SettleTime	This integer indicates the desired “quiet” time in 1/1000 seconds.

Return parameters

Result code	
1 or ATM_SUCCESS	The event was added.
-1 or ATM_NOTCONNECTED	The session is not connected.
-9 or ATM_SYSTEMERROR	A system error occurred.
-301 or ATM_EVENTALLREADYSET	The event ID supplied is already in use by this application for this table ID.
-302 or ATM_EVENTMAXEXCEEDED	The maximum number of events allowed for a table has already been added to this table.
-303 or ATM_TABLEMAXEXCEEDED	The maximum number of tables for a hWnd has already been used.

ATMAllowUpdates

The ATMAllowUpdates function allows your host access software to update the window display area when the host sends a message.

The ATMAllowUpdates function may be used with 3270 and 5250 sessions only. Emulator software ordinarily updates the session presentation space when host updates are received, which means ‘allow updates” is the default.

ATMAllowUpdates is used in conjunction with the ATMBlockUpdates function to control when the effect of host messages are displayed and is used after the ATMBlockUpdates function has been called.

Prerequisites

ATMConnectSession must be called before using this function.

Syntax

```
rc = ATMAllowUpdates ( hWnd)
```

Call parameters

Each call parameter of the ATMAllowUpdates function is described below.

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.

Return parameters

Result code	
1 or ATM_SUCCESS	Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-9 or ATM_SYSTEMERROR	A system error occurred.
-11 or ATM_RESOURCEUNAVAILABLE	Resource unavailable.
-104 or ATM_NOEMULATORATTACHED	No host access software attached.
-105 or ATM_WSCTRLFAILURE	Workstation Control failure.

ATMBlockUpdates

The ATMBlockUpdates function prevents the connected host session from updating the session's window display area when the host sends a message.

The ATMBlockUpdates function may be used with 3270 and 5250 sessions only.

Note Because the presentation space is a virtual display area, it is updated even if a call is made to ATMBlockUpdates. However, these updates are not visible to the user.

Prerequisites

ATMConnectSession must be called before using this function.

Syntax

```
rc= ATMBlockUpdates (hWnd)
```

Call parameters

Each call parameter of the ATMBlockUpdates function is described below.

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.

Return parameters

Result code

1 or ATM_SUCCESS	Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-9 or ATM_SYSTEMERROR	A system error occurred.
-11 or ATM_RESOURCEUNAVAILABLE	Resource unavailable.
-104 or ATM_NOEMULATORATTACHED	No host access software attached.
-105 or ATM_WSCTRLFAILURE	Workstation Control failure.

ATMClearEventTable

The ATMClearEventTable clears all events from the table specified. It can then be reloaded with new events.

In a complex application dealing with numerous large tables, memory can be conserved by clearing a table if it is no longer going to be used.

Prerequisites

ATMConnectSession must be called before using this function.

Syntax

```
rc = ATMClearEventTable ( hWnd,table)
```

Call parameters

Each call parameter of the ATMClearEventTable function is described below.

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
table	This integer indicates which table of events will be cleared.

Return parameters

Result code

1 or ATM_SUCCESS	The table was cleared.
-1 or ATM_NOTCONNECTED	The session is not connected.
-9 or ATM_SYSTEMERROR	A system error occurred.
-303 or ATM_TABLEMAXEXCEEDED	The supplied table number is greater than the maximum allowed.

ATMConnectSession

The ATMConnectSession function specifies the host session with which your application will interact.

Prerequisites

The session named in the ATMConnectSession call must previously have been assigned this value as its session short name, or PSID.

Syntax

```
rc = ATMConnectSession (hWnd, session)
```

Call parameters

Each call parameter of the ATMConnectSession function is described below.

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
session	A string containing the short name of the target host session.

Return parameters

Result code

1 or ATM_SUCCESS	Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-5 or ATM_SESSIONLOCKED	Session locked.
-9 or ATM_SYSTEMERROR	A system error occurred.
-200 or ATM_NOMATCHINGPSID	No matching Presentation Space ID (PSID).

ATMDeleteEvent

The ATMDeleteEvent function deletes the event identified by a hWnd, table ID, and event ID. That event ID can then be reused by adding another event with that event ID.

Prerequisites

ATMConnectSession must be called before using this function.

Syntax

```
rc = ATMDeleteEvent (hWnd,table,event)
```

Call parameters

Each call parameter of the ATMDeleteEvent function is described below.

Paramete	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
table	This integer indicates the table of events that the event is being deleted from.
event	The identifying integer that represents the event that is being deleted.

Return parameters

Result code

1 or ATM_SUCCESS	The event was deleted.
-1 or ATM_NOTCONNECTED	The session is not connected.
-9 or ATM_SYSTEMERROR	A system error occurred.
-302 or ATM_EVENTMAXEXCEEDED	The maximum number of events allowed for a table has already been added to this table.
-303 or ATM_TABLEMAXEXCEEDED	The maximum number of tables for a hWnd has already been used.
-305 or ATM_ INDEXNOTSET	A supplied event index has not been set.

ATMDisconnectSession

The ATMDisconnectSession function releases the previously connected host session.

Prerequisites

ATMConnectSession must be called before using this function.

Syntax

```
rc = ATMDisconnectSession ( hWnd)
```

Call parameters

Each call parameter of the ATMDisconnectSsisson function is described below.

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.

Return parameters

Result code

1 or ATM_SUCCESS	Successful.
-1 or ATM_NOTCONNECTED	Not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-9 or ATM_SYSTEMERROR	A system error occurred.
-200 or ATM_NOMATCHINGPSID	No matching Presentation Space ID (PSID).

ATMGetATMAPIVersion

The ATMGetATMAPIVersion function retrieves current version number of the ATMAPI32 DLL on the current system.

Syntax

```
rc = ATMGetATMAPIVersion ( buffer, bufferLength)
```

Call parameters

Each call parameter of the ATMExecute function is described below.

Parameter	Description
rc	The integer return value.
buffer	A string provided by the client. The current version number of the ATMAPI.DLL is copied into this destination. The return string uses the following format: " X.XX"
bufferLength	This integer represents the number of characters supplied in the buffer string. The minimum length of buffer is 5.

Return parameters

Result code

1 or ATM_SUCCESS	Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.

ATMGetConnectionStatus

The ATMGetConnectionStatus function returns status codes describing the current condition of the host connection.

Note Connection and disconnection of EAL functions are not the same as asynchronous emulation connect and disconnect. For example, async terminal sessions require that the emulator connect to and disconnect from a host.

Syntax

```
rc = ATMGetConnectionStatus ( hWnd, StatusType)
```

Call parameters

Each call parameter of the ATMGetConnectionStatus function is described below.

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
StatusType	An integer or constant that specifies the type of connection information you are requesting, as follows: 1 or ATM_XSTATUS -- Returns an integer indicating the status of the XCLOCK . Not applicable for async sessions. 2 or ATM_CONNECTION -- Returns an integer indicating the status of the host connection. 3 or ATM_ERROR -- Returns the integer value of any existing error condition. Not applicable for async sessions.

Return parameters

Result codes for StatusType ATM_XSTATUS

0	None of the following ATM_XSTATUS conditions hold
1 or ATM_INVALIDNUM X¥#	User entered an invalid number.
2 or ATM_NUMONLY X¥Num	User entered non-numeric data in a numeric field.
3 or ATM_PROTFIELD X<¥>	User typed in a protected field.
4 or ATM_PASTEOF X¥>	User typed past the end of a field.
5 or ATM_BUSY X( )	Host is busy.
6 or ATM_INVALIDFUNC X-f	Function is invalid.
7 or ATM_UNAUTHORIZED X¥X	Unauthorized printer.
8 or ATM_SYSTEM XSYSTEM	System locked during processing.
9 or ATM_INVALIDCHAR X-s	Invalid character entered.

Result codes for StatusType ATM_CONNECTION:

0	None of the following ATM_CONNECTION states hold
1 or ATM_APPOWNED 4B_	Session is connected to an application.
2 or ATM_SSCP 4B	Control program owns session.
3 or ATM_UNOWNED 4B?	Session not connected to an application.
4 or ATM_NONE 4B	None of the foregoing conditions applies. For async, not all connection statuses apply.
5 or ATM_UNKNOWN 4	

Result codes for StatusType ATM_ERROR:

0	None of the following ATM_ERROR states apply
1 or ATM_PROGCHECK XPROG756	Configuration mismatch occurred.
2 or ATM_COMMCHECK -+Z_510	Communications hardware problem occurred.
3 or ATM_MACHINECHECK x0211	Problem with the physical connection occurred.

ATMGetCursorLocation

The ATMGetCursorLocation function returns an integer that represents the current presentation space position of the cursor.

The value returned by this function is a presentation space position. To get the exact row or column position, call the ATMRowColumn function using the result of ATMGetCursorLocation as returned. The upper left corner of the presentation space is located at 1,1.

Prerequisites

ATMConnectSession must be called before using this function.

Syntax

```
rc = ATMGetCursorLocation ( hWnd)
```

Call parameters

Each call parameter of the ATMGetCursorLocation function is described below.

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.

Return parameters

Result code	
>0	Presentation space location of the cursor in the connected host session. Successful.
-1 or ATM_NOTCONNECTED	Not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-9 or ATM_SYSTEMERROR	A system error occurred.

ATMGetEmulatorPath

The ATMGetEmulatorPath function returns the full path of your user's directory. This function is useful when you need to find the programs and configurations for your host access software.

Syntax

```
rc = ATMGetEmulatorPath (hWnd, buffer, bufferLength)
```

Call parameters

Each call parameter of the ATMGetEmulatorPath function is described below.

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
buffer	A string that is used as the destination of the returned path. The string must be large enough to accommodate a fully qualified system path plus one character.
bufferLength	This integer determines the number of characters that are allocated for buffer. The maximum size for buffer is 128 characters.

Return parameters

Result code	
>0	The number of characters in buffer Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-9 or ATM_SYSTEMERROR	A system error occurred.
-105 or ATM_WSCTRLFAILURE	Workstation Control failure.

Comments

The following is an example of the value returned in buffer by this function:

```
C:\Program Files\Attachmate\E!E2K
```

ATMGetEmulatorVersion

The ATMGetEmulatorVersion function returns the version number of the executable file for your host access software.

Syntax

```
rc = ATMGetEmulatorVersion (hWnd, buffer, bufferLength)
```

Call parameters

Each call parameter of the ATMGetEmulatorVersion function is described below.

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
buffer	A string at least 6 characters long that is used as the destination for the returned version number. This string is returned in the following format: " X.XX"
bufferLength	This integer determines the number of characters allocated for buffer.

Return parameters

Result code	
>0	The number of characters in buffer Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-9 or ATM_SYSTEMERROR	A system error occurred.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.

ATMGetError

The ATMGetError function returns the name of the function in which the most recent function call error occurred. ATMGetError also provides a description of the error.

Prerequisites

A successful call to ATMConnectSession must have been made prior to using this function.

Syntax

```
rc = ATMGetError ( hWnd, buffer, bufferLength)
```

Call parameters

Each call parameter of the ATMGetError function is described below.

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
buffer	A string that is used as the destination for the function name and error description. This string should be large enough to accommodate at least 121 characters.
bufferLength	This integer determines the number of characters that are allocated for buffer.

Return parameters

Result code	
>0	The number of characters in buffer Successful.
0	No errors detected.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.
-104 or ATM_NOEMULATORATTACHED	No terminal session is loaded.

Comments

Information returned in buffer is in two parts: the function name and the description of the error, if an error is detected. The two portions of the string are delimited by a new line (NL) character.

ATMGetFieldInfo

The ATMGetFieldInfo function indicates whether the specified field matches the desired criteria. This function cannot be used with async sessions.

Prerequisites

A successful call to ATMConnectSession must have been made prior to using this function.

Syntax

```
rc = ATMGetFieldInfo ( hWnd, row,column, infoType)
```

Call parameters

Each call parameter of the ATMGetFieldInfo function is described below.

Parameter	Description	
rc	The integer return value.	
hWnd	This integer indicates the window handle of your application.	
row	An integer or integer expression corresponding to the row position on the connected session containing the field.	
column	An integer or integer expression corresponding to the column position on the connected session containing the field.	
infoType	Specifies which of the following conditions to check:	
	1 or ATM_ISFIELDPROTECTED	Checks for protected status.
	2 or ATM_ISFIELDNUMERIC	Checks whether the field is numeric.
	3 or ATM_ISFIELDSELECTORPENDETECTABLE	Checks whether the field is light pen selectable.
	4 or ATM_ISFIELDBOLD	Checks whether the field is bold.
	5 or ATM_ISFIELDHIDDEN	Checks whether the field is hidden.
	6 or ATM_ISFIELDMODIFIED	Checks whether the field has the modified flag set. (Some host applications set the modified flag at all times.)

Return parameters

Result code	
1 or ATM_SUCCESS	Field matches criteria.
0 or ATM_NOTATTRIBUTE	Field does not match criteria.
-1 or ATM_NOTCONNECTED	The session is not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameters exist.
-5 or ATM_SESSIONLOCKED	Session locked.
-7 or ATM_INVALIDPOSITION	Presentation space position is invalid.
-9 or ATM_SYSTEMERROR	A system error occurred.

Tip

The ATMGetFieldInfo function can also be used to determine a field's color. By default, 3270 host access software uses the color scheme as follows

If the field is	And	The default color is
Protected	Not bold	Cyan
Protected	Bold	White
Unprotected	Not bold	Green
Unprotected	Bold	Red

ATMGetFieldLength

The ATMGetFieldLength function searches through the current host session for the beginning position of a specified field. ATMGetFieldLength returns the length of the field, if the field is located. This function can return the length of the following:

- The current field
- The first field following a specified position
- The field preceding a specified position
- The next protected field in relation to a specified position
- The previous protected field in relation to a specified position
- The first protected field on the screen
- The first unprotected field on the screen

Prerequisites

A successful call to ATMConnectSession must have been made prior to using this function.

Syntax

```
rc = ATMGetFieldLength (hWnd, row, column, type)
```

Call parameters

Each call parameter of the ATMGetFieldLength function is described below.

Parameter	Description	
rc	The integer return value.	
hWnd	This integer indicates the window handle of your application.	
row	This integer defines the beginning row position of the target field of the current host session.	
column	This integer defines the beginning column position of the target field of the current host session.	
type	This integer indicates the type of field requested. Valid values are as follows:	
	ATM_THISFIELD or 0	Returns length of the field located at row and column.
	ATM_NEXTFIELD or 1	Returns the length of the field following the field located at row and column.
	ATM_PREVIOUSFIELD or 2	Returns length of field directly before the field located at row and column.
	ATM_NEXTPROTECTEDFIELD or 3	Returns length of next protected field in relation to row and column.
	ATM_NEXTUNPROTECTEDFIELD or 4	Returns length of next unprotected field in relation to row and column.
	ATM_PREVIOUSPROTECTEDFIELD or 5	Returns length of previous protected field in relation to row and column.
	ATM_PREVIOUSUNPROTECTEDFIELD or 6	Returns length of previous unprotected field in relation to row and column.

Return parameters

Result code

>0	Field length Successful.
0	Field of type could not be located.
-1 or ATM_NOTCONNECTED	The session is not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameters exist.
-7 or ATM_INVALIDPOSITION	Invalid presentation space position.
-9 or ATM_SYSTEMERROR	A system error occurred.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.

ATMGetFieldPosition

The ATMGetFieldPosition function searches through the current host session for the beginning position of a specified field. If the field is located, ATMGetFieldPosition returns an integer that defines the field's starting position relative to the upper left corner of the presentation space.

To get the exact row or column position, call the ATMRowColumn function using the result of ATMGetFieldPosition as input.

Prerequisites

A successful call to ATMConnectSession must have been made prior to using this function.

Syntax

```
rc = ATMGetFieldPosition ( hWnd,row,column,type)
```

Call parameters

Each call parameter of the ATMGetFieldPosition function is described below.

Parameter	Description	
rc	The integer return value.	
hWnd	This integer indicates the window handle of your application.	
row	This integer defines the beginning row position of the target field of the current host session.	
column	This integer defines the beginning column position of the target field of the current host session.	
type	This integer indicates the type of field requested. Valid values are as follows:	
	ATM_THISFIELD or 0	Returns length of the field located at row and column.
	ATM_NEXTFIELD or 1	Returns the length of the field following the field located at row and column.
	ATM_PREVIOUSFIELD or 2	Returns length of field directly before the field located at row and column.
	ATM_NEXTPROTECTEDFIELD or 3	Returns length of next protected field in relation to row and column.
	ATM_NEXTUNPROTECTEDFIELD or 4	Returns length of next unprotected field in relation to row and column.
	ATM_PREVIOUSPROTECTEDFIELD or 5	Returns length of previous protected field in relation to row and column.
	ATM_PREVIOUSUNPROTECTEDFIELD or 6	Returns length of previous unprotected field in relation to row and column.

Return parameters

Result code	
>0	Field position Successful.
0	Field of type could not be located.
-1 or ATM_NOTCONNECTED	The session is not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameters exist.
-7 or ATM_INVALIDPOSITION	Invalid presentation space position.
-9 or ATM_SYSTEMERROR	A system error occurred.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.

ATMGetKeystroke

The ATMGetKeystroke function allows your application to receive the keystrokes for the sessions specified with the StartKeystrokeIntercept function.

Use the SendKey function to pass both original keystrokes and any others to the target presentation space.

When keystrokes are available, they are read into your applications data area. Each keystroke is represented by a key code. For a list of all key mnemonics, refer to Appendix B, "PC-Host Keyboard Mnemonics."

Prerequisites

A successful call to ATMStartKeystrokeIntercept must have been made prior to using this function.

Syntax

```
rc = ATMGetKeystroke ( hWnd, Session,KeyBuffer,Bufferlength)
```

Call parameters

Each call parameter of the ATMGetKeystroke function is described below.

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
Session	The session short name.
KeyBuffer	The string that returns the key.
Bufferlength	This integer specifies the size of the buffer.

Return parameters

Result code	
0	Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameters exist.
-7 or ATM_INVALIDPOSITION	Invalid presentation space position.
-9 or ATM_SYSTEMERROR	A system error occurred.
-11 or ATM_RESOURCEUNAVAILABLE	Resource unavailable.
-102 or ATM_DELAYEDBYCLIENT	Delayed by client.

ATMGetLayoutName

The ATMGetLayoutName function returns the name of the currently loaded layout file. The layout file stores information about session setup, window organization, and other options.

Syntax

```
rc = ATMGetLayoutName ( hWnd,buffer, bufferLength)
```

Call parameters

Each call parameter of the ATMGetLayoutName function is described below.

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
buffer	A string that is used as the destination for the returned filename and path. This string must be large enough to accommodate a fully qualified system path plus one character.
bufferLength	This integer determines the number of characters that are allocated for buffer.

Return parameters

Result code	
>0	The number of characters in buffer Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameters exist.
-9 or ATM_SYSTEMERROR	A system error occurred.
-105 or ATM_WSCTRLFAILURE	Workstation Control failure.

ATMGetParameter

The ATMGetParameter function returns the current settings of your host system parameters indicated in the *paramIndex* parameter.

The parameters are set using the ATMSetParameter function. They can also be reset using the ATMResetSystem function, which reinitializes the parameters to their default values.

For a complete list of system parameters, settings, and corresponding index numbers, see "Appendix C, "System Parameter Settings."

Syntax

```
rc = ATMGetParameter ( hWnd, paramIndex)
```

Call parameters

Each call parameter of the ATMGetParameter function is described below.

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
paramIndex	This integer indicates the global parameter for which the value is wanted.

Return parameters

Result code

>0	Present setting of the parameter queried. Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameters exist.
-9 or ATM_SYSTEMERROR	A system error occurred.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.

ATMGetSessionHandle

The ATMGetSessionHandle function obtains the window handle of the specified host session. This window handle can be used to call Windows API functions.

Syntax

```
rc = ATMGetSessionHandle ( hWnd, session)
```

Call parameters

Each call parameter of the ATMGetSessionHandle function is described below.

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
session	A string containing the short name of the target host session.

Return parameters

Result code	
>0	The window handle of the specified host session. Successful.
-1 or ATM_NOTCONNECTED	Not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameters exist.
-9 or ATM_SYSTEMERROR	A system error occurred.
-11 or ATM_RESOURCEUNAVAILABLE	Resource unavailable.
-104 or ATM_NOEMULATORATTACHED	No terminal session is loaded.
-105 or ATM_WSCTRLFAILURE	Workstation Control failure.

ATMGetSessions

The ATMGetSessions function returns either a string containing a list of session names or an integer containing the number of sessions meeting a particular condition.

Using the *sessionState* parameter, an application can request information about sessions as follows:

- All configured sessions
- All opened sessions
- All powered sessions
- The number of configured sessions
- The number of opened sessions
- The number of powered sessions
- All emulated sessions
- All emulated sessions that are currently powered on
- The number of emulated sessions
- The number of emulated sessions that are currently powered on

Syntax

```
rc = ATMGetSessions(hWnd,sessionList,bufferLength,sessionState)
```

Call parameters

Each call parameter of the ATMGetSessions function is described below.

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
sessionList	A string provided by the client used as the destination for a comma-delimited string containing session names that meet the criteria defined by sessionState. The string must be large enough to accommodate the maximum length of the session name (10 characters) for each session matching the sessionState criteria.
bufferLength	An integer defining the length of the sessionList parameter.
sessionState	Defines one of the following options: 1 or ATM_GETCONFIGURED Create a list of configured sessions. 2 or ATM_GETOPENED Create a list of all opened sessions. 3 or ATM_GETPOWERED Create a list of all powered sessions. 4 or ATM_GETCONFIGUREDCOUNT Return the number of configured sessions. 5 or ATM_GETOPENEDCOUNT Return the number of opened sessions. 6 or ATM_GETPOWEREDCOUNT Return the number of powered sessions. 11 or ATM_GETEMULATED Create a list of emulated sessions. 12 or ATM_GETEMULATEDPOWERED Create a list of emulated sessions that are currently powered on. 14 or ATM_GETEMULATEDCOUNT Return the number of emulated sessions. 15 or ATM_GETEMULATEDPOWEREDCOUNT Return the number of emulated sessions that are currently powered on.

For the “count” options, the sessionList parameter is ignored and can be set to an empty string.

Return parameters

Result code

>0	The number of characters copied into sessionList for the following: ATM_GETCONFIGURED ATM_GETOPENED ATM_GETPOWERED ATM_GETEMULATED ATM_GETEMULATEDPOWERED Or the number of sessions meeting the session state criteria for the following: ATM_CONFIGUREDCOUNT ATM_OPENEDCOUNT ATM_POWEREDCOUNT ATM_EMULATEDCOUNT ATM_EMULATEDPOWEREDCOUNT Successful.
0	Unsuccessful (no sessions were found)
-2 or ATM_INVALIDPARAMETER	Invalid parameters exist. This may also indicate that the allocated buffer size is too small. Increase the buffer size.
-9 or ATM_SYSTEMERROR	A system error occurred.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.

ATMGetSessionSize

The ATMGetSessionSize function calculates the presentation space size of the connected host session. This function returns the total number of characters in the presentation space.

This function is useful for determining the model screen size to which the session is currently connected. The first presentation space position is the upper left corner of the screen, and the last position is the lower right corner. For example, if the terminal session is emulating a Model 2 screen with 24 rows and 80 columns, the total number of presentation space positions would be 24x80=1,920. Therefore, 1920 is the returned value. For example, the following 3270 values are returned (depending on model type):

- Model 2 (24x80) = 1,920 presentation space positions
- Model 3 (32x80) = 2,560 presentation space positions
- Model 4 (43x80) = 3,440 presentation space positions
- Model 5 (27x132) = 3,564 presentation space positions

Use the ATMRowColumn function to convert these values to row and column values.

Prerequisites

A successful call to ATMConnectSession must have been made prior to using this function.

Syntax

```
rc = ATMGetSessionSize ( hWnd)
```

Call parameters

Each call parameter of the ATMGetSessionsSize function is described below.

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.

Return parameters

Result code	
>0	The number of individual presentation space positions on the connected host session Successful.
-1 or ATM_NOTCONNECTED	Not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameters exist.
-9 or ATM_SYSTEMERROR	A system error occurred.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.
-104 or ATM_NOEMULATORATTACHED	No terminal session is loaded.

ATMGetSessionStatus

The ATMGetSessionStatus function determines whether a target session meets the specified status condition.

Syntax

```
rc = ATMGetSessionStatus ( hWnd, session, sessionType)
```

Call parameters

Each call parameter of the ATMGetSessionsStatus function is described below.

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
session	A string containing the short name of the target host session.
sessionType	This integer specifies the session status being requested. Valid values and descriptions are: 1 or ATM_ISCONFIGURED The target session is configured. 2 or ATM_ISOPENED The target session is open. 3 or ATM_ISPOWERED The target session is currently powered on. 14 or ATM_ISEMULATED The target session has a terminal window loaded. 15 or ATM_ISCONNECTED The target session is currently connected. 16 or ATM_ISFILETRANSFER The target session is performing a file transfer.

Return parameters

Result code	
0	Unsuccessful (condition is false).
1 or ATM_SUCCESS	Successful (condition is true).
-1 or ATM_NOTCONNECTED	Not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameters exist.
-9 or ATM_SYSTEMERROR	A system error occurred.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.
-200 or ATM_NOMATCHINGPSID	No matching Presentation Space ID (PSID).

ATMGetString

The ATMGetString function copies a string from the connected session into a string provided by the application program.

Prerequisites

A successful call to ATMConnectSession must have been made prior to using this function.

Syntax

```
rc = ATMGetString ( hWnd, row,column, buffer, bufferLength)
```

Call parameters

Each call parameter of the ATMGetString function is described below.

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
row	This integer indicates the starting row position of the string to copy from the host session.
column	This integer indicates the starting column position of the string to copy from the host session.
buffer	The destination string that receives the copied string from the target session. It must be large enough to handle the number of characters specified in the bufferLength parameter.
bufferLength	An integer that defines the number of characters to copy, beginning from the position specified in the row and column parameters.

The function stops copying when it reaches the last character specified in the *bufferLength* parameter.

Use the ATMSetParameter function to specify whether data passed between the PC and host contains field and character extended attribute bytes (EABs). Specifying 1 yields 2 characters for each session character. Therefore, if you have set the EAB parameter to 1, you should allow for twice as many characters. If the parameter is set to 2 (default), data will be passed without EABs.

EABs are not available with async sessions

Return parameters

Result code	
1 or ATM_SUCCESS	Successful.
-1 or ATM_NOTCONNECTED	Not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameters exist.
-4 or ATM_TIMEOUT	Time-out expired.
-5 or ATM_SESSIONLOCKED	Session locked.
-7 or ATM_INVALIDPOSITION	Invalid presentation space position.
-9 or ATM_SYSTEMERROR	A system error occurred.

ATMGetStringFromField

The ATMGetStringFromField function copies all characters from a specified field in the connected host session to a string provided by the client. This function can be used for either protected or unprotected fields, but the host screen must be field-formatted.

Prerequisites

You must call ATMConnectSession before using this function.

Syntax

```
rc = ATMGetStringFromField(hWnd,row,column,buffer,bufferLength)
```

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
row	This integer indicates the starting row position of the string to copy from the host session.
column	This integer indicates the starting column position of the string to copy from the host session.
buffer	The destination string that receives the copied string from the target session. It must be large enough to handle the number of characters specified in the bufferLength parameter.
bufferLength	An integer that defines the number of characters to copy, beginning from the position specified in the row and column parameters.

Return Parameters

Result Code	
1 or ATM_SUCCESS	Successful.
-1 or ATM_NOTCONNECTED	Not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameters exist.
-5 or ATM_SESSIONLOCKED	Session locked.
-6 or ATM_DATATRUNCATED	Data truncated.
-7 or ATM_INVALIDPOSITION	Invalid presentation space position.
-9 or ATM_SYSTEMERROR	A system error occurred.
-24 or ATM_UNFORMATTEDHOSTPS	Unformatted host Presentation Space ID.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.

ATMHoldHost

The ATMHoldHost function stops data stream input from the host from changing the session's emulated screen. In contrast, ATMBlockUpdates stops only the repainting of changes made to the screen. With ATMBlockUpdates, the data from the host does change the presentation space, but it is not shown on the window of the emulated screen. With ATMHoldHost, the screen updates are held back.

Prerequisites

An ATMConnectSession is required before this function.

Syntax

```
rc = ATMHoldHost ( hWnd)
```

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.

Return Parameters

Result Code	
1 or ATM_SUCCESS	Successful.
-1 or ATM_NOTCONNECTED	Not connected.
-9 or ATM_SYSTEMERROR	A system error occurred.

Comments

ATMHostHost can be used only with async sessions.

If this function is not issued, there is a small window of time between sending a key or string to the host and the establishment of an ATMWaitForEvent. If this time allows the event to occur and pass, it may be missed and not be reported by ATMWaitForEvent. (However, it does not hold for an idle period that is being waited for.)

ATMWaitForEvent performs an internal ATMResumeHost when ready to trap events.

ATMListSessions

The ATMListSessions function displays a list box containing the sessions that meet a specified condition in the current configuration. The user can select one session from the list. The selected session is returned by the function.
The available session conditions are:

1 = List sessions that are configured
2 = List sessions that are open
3 = List sessions that are powered
11 = List emulated sessions
12 = List sessions that are both emulated and powered

Prerequisites

None.

Syntax

```
rc = ATMListSessions ( hWnd,buffer, bufferLength, title, type)
```

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
buffer	A string used as the destination should be at least 11 characters long. The returned string uses the format: `"session_short_name session_long_name"`
bufferLength	An integer that defines the length of the buffer parameter.
title	An optional string passed by the client that contains a message that will be attached to the list box. By default, this title is “Choose a Session.” title must be 30 or fewer characters in length. To send the default string, set title to an empty string (`""`).
type	An integer defining the session status type you want to the list box. The type parameter constants have the following values and descriptions: 1 or ATM_GETCONFIGURED List configured sessions. 2 or ATM_GETOPENED List opened sessions. 3 or ATM_GETPOWERED List powered sessions. 11 or ATM_GETEMULATED List emulated sessions. 12 or ATM_GETEMULATEDPOWERED List sessions that are both emulated and powered.

Return Parameters

Result Code	
>0	Number of matching sessions available.
0	No sessions of that type available.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.

ATMLockKeyboard

The ATMLockKeyboard function prevents a user from typing in characters from the keyboard in the connected session. Note that other programs can still send characters.

Caution If this function is called in your application, make sure your application uses ATMUnlockKeyboard to unlock the keyboard before exiting.

Prerequisites

You must call ATMConnectSession before using this function.

Syntax

```
rc = ATMLockKeyboard ( hWnd)
```

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.

Return Parameters

Result Code	
1 or ATM_SUCCESS	Successful.
-1 or ATM_NOTCONNECTED	Not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-9 or ATM_SYSTEMERROR	A system error occurred.

Comments

Use ATMUnlockKeyboard to unlock the keyboard.

ATMOpenLayout

The ATMOpenLayout function opens an EXTRA! layout file.

Prerequisites

None.

Syntax

```
rc = ATMOpenLayout (hWnd, layout)
```

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
layout	A string containing the path and name of the host access layout file.

Return Parameters

Result Code	
1 or ATM_SUCCESS	Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-9 or ATM_SYSTEMERROR	A system error occurred.
-105 or ATM_WSCTRLFAILURE	Workstation Control failure.

ATMPause

The ATMPause function waits for a specified amount of time in half-second increments. If the host response is expected to be delayed, you can use this function to temporarily halt the application.

Caution Use ATMPause with caution. When waiting for a host response, it is more efficient to call ATMWaitForString, ATMWaitForCursor, or ATMWaitHostQuiet.

Prerequisites

None.

Syntax

```
rc = ATMPause ( hWnd, time)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
time	An integer that defines the number of half-second increments to pause. For example, 1 = half-second, 2 = one second, and 10 = five seconds.

Return Parameters

Result code	
1 or ATM_SUCCESS	Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-9 or ATM_SYSTEMERROR	A system error occurred.
-11 or ATM_RESOURCEUNAVAILABLE	Resource unavailable.
-26 or ATM_HOSTSESSIONUPDATE	Host session update.
-102 or ATM_DELAYEDBYCLIENT	Delayed by client.

Comments

The ATMPause function yields to other Windows applications while it is pausing your application. Unlike the ATMWait function, ATMPause **does not** monitor the Operator Information Area (OIA).

ATMReceiveFile

The ATMReceiveFile function transfers a file from the host to the PC. A file transfer from the host requires different options depending on the host's operating system: CMS, TSO, or CICS.

Prerequisites

The target session for the file transfer must be ready for the file transfer: the login has been performed, the keyboard is unlocked, the session is not engaged in another file transfer, and so on. Typically, the host must be in the native operating system at a "Ready" prompt before a file transfer can be run.

Syntax

```
rc = ATMReceiveFile ( hWnd, command, commandSize)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
command	A string containing the appropriate command and option parameters. See Appendix C, "File Transfer Command Parameters," for additional information.
commandSize	The integer length of the command string parameter.

Return Parameters

Result Code	
1 or ATM_SUCCESS	Successful.
-1 or ATM_NOTCONNECTED	Not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-4 or ATM_TIMEOUT	Time-out expired.
-7 or ATM_INVALIDPOSITION	Invalid position.
-9 or ATM_SYSTEMERROR	A system error occurred.
-11 or ATM_RESOURCEUNAVAILABLE	Resource unavailable.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.
-102 or ATM_DELAYEDBYCLIENT	Delayed by client.

ATMRegisterClient

The ATMRegisterClient function establishes the type of host access (EXTRA!, KEA!, Irma, RALLY!, INFOConnect, or HP) that will use this window handle for subsequent function calls. It must be the **first** EAL call issued in the application.

Prerequisites

None.

Syntax

```
rc = ATMRegisterClient ( hWnd,emulatorType)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
emulatorType	This integer indicates the type of host access to use.

Return Parameters

Result code	
1 or ATM_SUCCESS	Registration is complete.
-9 or ATM_SYSTEMERROR	A system error occurred.

Comments

ATMRegisterClient must be the first EAL function call within your application. If this call is not issued, the host access type defaults to EXTRA!, allowing existing code to run unmodified. However, it is recommended that you register and unregister for EXTRA! as well as other host access software, to allow cleanup to occur at the time of unregistration. This will result in an error if EXTRA! is not installed on your system.
When using host access values, use the defined names rather than the integer values. Host access values are:

- ATM_EXTRA = 1 (EPC in 32-bit; EXTRA! in 16-bit)
- ATM_RALLY = 2
- ATM_KEA = 3 (KEA! 95 in 32-bit; KEA! for VT in 16-bit)
- ATM_IRMA = 4
- ATM_ICONN = 5
- ATM_HP = 6 (KEA! 700/98 for HP)

ATMResetSystem

The ATMResetSystem function reinitializes system parameters to their starting state and disconnects any previously connected sessions. This function also resets each session parameter to its default value and clears any event tables that may be present.

Prerequisites

None.

Syntax

```
rc = ATMResetSystem (hWnd)
```

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.

Return Parameters

Result code	
1 or ATM_SUCCESS	Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-9 or ATM_SYSTEMERROR	A system error occurred.

Comments

Your application can call this function at any time to reset the session values. Your application should call this function before exiting. For more information about system parameters, refer to "Appendix B, "System Parameter Settings." The ATMSetParameter function is used to set the system parameters.

ATMResumeHost

The ATMResumeHost function allows data stream input from the host to change the session's emulated screen, allowing all the host data sent since the last hold to be played onto the session screen. All screens between the screen preceding the ATMHoldHost and the current screen are included.
In contrast, the ATMAllowUpdates function only repaints the current state of the session to make it visible. Between an ATMBlockUpdates and ATMAllowUpdates, many screen changes may have been sent by the host. All of these are skipped when the final state of the screen is painted.

Prerequisites

An ATMConnectSession is required before this function.

Syntax

```
rc = ATMResumeHost (hWnd)
```

Table 5-92. ATMResumeHost parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.

Comments

ATMResumeHost can be used only with async sessions.
If the host has not been held, this function will report success without changing anything.

When it is ready to trap events, ATMWaitForEvent performs an internal ATMResumeHost (if ATMHoldHost has been used) to allow the events to occur on the screen. If required, it then reestablishes the ATMHoldHost.

ATMRowColumn

The ATMRowColumn function converts a presentation space position to a row or column position depending on the selection in the *option* parameter. Call this function twice if you need both the row and column position. This function is useful subsequent to a call to ATMGetCursorLocation or ATMSearchSession in which the presentation space position is returned.

Prerequisites

You must call ATMConnectSession before using this function.

Syntax

```
rc = ATMRowColumn ( hWnd,psPosition,option)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
psPosition	An integer representation of the presentation space position of the session connected to the calling application.
option	This integer indicates whether row or column will be returned. option parameter constants have the following values and descriptions: 0 or ATM_GETROW Converts the psPosition parameter to a row position. 1 or ATM_GETCOLUMN Converts the psPosition parameter to a column position.

Return Parameters

Result code	
>0	Success: the column or row position for the connected host session.
-1 or ATM_NOTCONNECTED	Not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-7 or ATM_INVALIDPOSITION	Invalid position.
-9 or ATM_SYSTEMERROR	A system error occurred.

Comments

The size of the presentation space depends on your screen model, which is determined by your session configuration. For example, if a session is configured to display 24 rows by 80 columns, the presentation space contains 1,920 separate, unique character addresses, or presentation space positions. The positions start from the top left corner of the presentation space and continue to the bottom right. In a session configured as described above, a presentation space position of 84 would result in row 2, column 4.

ATMRunEmulatorMacro

The ATMRunEmulatorMacro function runs a previously created host access macro. Host access macros can be useful for the following tasks:

- Logging in to the host
- Logging off the host
- Automating file transfers

Prerequisites

None.

Syntax

```
rc = **ATMRunEmulatorMacro** ( hWnd,session,macroName)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
session	A string containing the target session's short name.
macroName	A string containing the path and name of the macro that you want to run.

Return Parameters

Result Code	
1 or ATM_SUCCESS	Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.

Comments

The macro must have been created in your host access software in order to run successfully. Refer to the user's guide for your host access software for details.

ATMSearchField

The ATMSearchField function searches the specified field in the connected presentation space for a string. The string can reside in either protected or unprotected fields of a field-formatted host presentation space.

Prerequisites

You must call ATMConnectSession before using this function.

Syntax

```
rc = ATMSearchField (hWnd, row, column, searchString)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
row	This integer defines the beginning row position within the field to search in the connected host session.
column	This integer defines the beginning column position within the field to search in the connected host session.
searchString	The string you want to locate.

Return Parameters

Result Code	
>0	Success: the beginning presentation space position where the string was located.
0 or ATM_NOTFOUND	Unsuccessful (the string could not be located).
-1 or ATM_NOTCONNECTED	Not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-7 or AATM_INVALIDPOSITION	Invalid presentation space position.
-9 or ATM_SYSTEMERROR	A system error occurred.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.

Comments

If the target string is found, this function returns its starting position in the presentation space. Use the ATMRowColumn function to determine the row or column position for the presentation space position returned.
This function is affected by the *searchOption* parameter defined in the ATMSearchSession function.

ATMSearchSession

The ATMSearchSession function provides various ways to search through the current session for a specific string. These search options are available:

- Search the entire host screen of the connected session
- Begin searching at the specified location and return the position of the first occurrence of the string
- Search for the string at a specified location only
- Search the connected session starting at the location specified and searching from the bottom right to the top left of the target session.

Prerequisites

You must call ATMConnectSession before using this function.

Syntax

```
rc = ATMSearchSession(hWnd,row,column,searchString,searchOption)
```

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
row	This integer defines the beginning row position to search in the connected host session.
column	This integer defines the beginning column position to search in the connected host session.
searchString	The string you want to locate.
searchOption	This integer indicates which type of search to perform. Available options are: 1 or ATM_SEARCHALL Searches the entire screen of the connected session. 2 or ATM_SEARCHFROM Searches from the specified the row and column parameters and returns the position of the first occurrence of the string, if found. 3 or ATM_SEARCHAT Searches at the specified row and column location only. 4 or ATM_SEARCHBACK Searches the connected session starting from the specified row and column parameters, or from the bottom right to the top left of the target session, if row and column are zero. **Note** If either or both row or column is set to zero, the search will begin at the last position of the target session.

Return Parameters

Result Code
For options 1, 2, or 4, the return value is the presentation space position of the string.
For option 3, 1 indicates that the string was found.

0 or ATM_NOTFOUND	Unsuccessful (the string could not be located).
-1 or ATM_NOTCONNECTED	Not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-5 or ATM_SESSIONLOCKED	Session locked.
-7 or ATM_INVALIDPOSITION	Invalid presentation space position.
-9 or ATM_SYSTEMERROR	A system error occurred.

ATMSendAndWait

The ATMSendAndWait function combines two common tasks—it sends a keystroke and searches the next host screen for a particular string. The host keystroke you send can be preceded by a text string. After sending the keystroke, the function will continuously search the host screen until it finds the specified search string or until it reaches a time-out period. The search string should be a unique string that appears on the next screen.

Prerequisites

You must call ATMConnectSession before using this function.

Syntax

```
rc = ATMSendAndWait ( hWnd, row, column, sendString,
                  waitString, timeOut)
```

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
row	This integer defines the row position at which waitString will be found in the connected host session.
column	This integer defines the column position where waitString will be found in the connected host session.
sendString	The string you want to send to the host. sendString can be text followed by an AID key. Refer to “Appendix D, “PC-Host Keyboard Mnemonics,” for a list of host keystrokes.
waitString T	he string you want to appear on the host after sendString has been sent.
timeOut	This integer defines the number of half-seconds to wait for the waitString to appear at the row and column location before timing out.

Comments

The search string should be a unique string that appears on the next screen.

Return Parameters

Result Code	
1 or ATM_SUCCESS	Successful.
0 or ATM_NOTFOUND	Unsuccessful (the string could not be located).
-1 or ATM_NOTCONNECTED	Not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-5 or ATM_SESSIONLOCKED	Session locked.
-7 or ATM_INVALIDPOSITION	Invalid presentation space position.
-9 or ATM_SYSTEMERROR	A system error occurred.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.

ATMSendFile

The ATMSendFile function transfers a file from the PC to the host session. A file transfer from the PC requires different options depending on the host operating system: CMS, TSO, or CICS.

Prerequisites

The target session for the file transfer must be ready. The login should have been performed, the keyboard should be unlocked, and the session should not be engaged in another file transfer. Usually the host must be in the native operating system at a "Ready" prompt before a file transfer can be run. See Appendix C, "File Transfer Command Parameters," for more information about file transfers.

Syntax

```
rc = ATMSendFile ( hWnd, command, commandSize)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
command	A string containing the appropriate command and option parameters.
commandSize	This integer defines the length of the command string parameter.

Return Parameters

Result Code	
1 or ATM_SUCCESS	Successful.
-1 or ATM_NOTCONNECTED	Not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-4 or ATM_TIMEOUT	A time-out occurred.
-7 or ATM_INVALIDPOSITION	Invalid presentation space position.
-9 or ATM_SYSTEMERROR	A system error occurred.
-11 or ATM_RESOURCEUNAVAILABLE	Resource unavailable.
-101 or AATM_MEMORYUNAVAILABLE	Memory unavailable.

ATMSendKey

This function allows you to use ASCII characters to represent the special functions of a host keystroke. Use ATMSendKey to send characters and host keystrokes to the connected host session from your application as if the user typed the characters on the keyboard.
With KEA!, ATMSendKey supports the key codes that are common to the 3270 and VT keyboards. If you want to send a keystroke that is not supported, you can send the ANSI numeric representation of the keystroke. For example, in Visual Basic the Chr$ function returns a single character containing the supplied number. Thus Chr$(3) is equal to CTRL C. If you concatenate this on the end of the string sent with ATMSendKey, it will be passed straight through to the host.
If you are used to KEA! macros, the caret (^) representations will not work. That is, ^M will be passed to the host as 2 characters—not as a carriage return code. Use @E, @N, or Chr$ (13) instead.

Prerequisites

You must call ATMConnectSession before using this function.

Syntax

```
rc = ATMSendKey ( hWnd, keys)
```

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
keys	A string containing the host control characters to be sent to the host session.

Return Parameters

Return value	Indication
1 or ATM_SUCCESS	Successful.
-1 or ATM_NOTCONNECTED	Not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-4 or ATM_SESSIONOCCUPIED	Session occupied.
-5 or ATM_SESSIONLOCKED	Session locked.
-9 or ATM_SYSTEMERROR	A system error occurred.

Comments

ATMSendKey has a 256 byte limitation. Exceeding this will result in the return value -2, Invalid parameter.
This function allows you to send both characters and host AID keys like Enter, PF1, Reset, and so forth. Note that you can include only one AID key per function call, and it must appear at the end of the string passed to the function. After calling ATMSendKey and before sending the next keystroke, you should call one of the wait functions (which include ATMWaitForString, ATMWaitForCursor, or ATMWaitHostQuiet).
Refer to Appendix D, “PC-Host Keyboard Mnemonics,” for a list of AID keys and host key codes.

ATMSendString

The ATMSendString function sends a string from the application into the connected host session at the presentation space position specified by the *row* and *column* parameters.
This function is similar to ATMSendKey, but can send only character strings, not host AID keys. Since the data is not sent as keystrokes, this function proves to be the more efficient choice when sending large amounts of data to a host screen.

Prerequisites

You must call ATMConnectSession before using this function.

Syntax

```
rc = ATMSendString (hWnd, row, column, string)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
row	This integer defines the beginning row position to send the string within the connected host session. If row is set to zero, the string will be sent to the current cursor position.
column	This integer defines the beginning column position to send the string within the connected host session. If column is set to zero, the string will be sent to the current cursor position. string The string to be copied to the host session.

Return Parameters

Return value Indication
1 or ATM_SUCCESS Successful.
-1 or ATM_NOTCONNECTED Not connected.
-2 or ATM_INVALIDPARAMETER Invalid parameter.
-5 or ATM_SESSIONLOCKED Session locked.
-7 or ATM_INVALIDPOSITION Invalid presentation space position.
-9 or ATM_SYSTEMERROR A system error occurred.

Comments

The row and column parameters are not available for async sessions. The string is sent to the current cursor position only. These parameters must be entered, but any value is accepted and ignored.

If either or both of the row and column parameters are zero, the string will be copied to the present host-cursor position. You can query the cursor location by using the ATMGetCursorLocation function before sending the string.

ATMSendStringToField

The ATMSendStringToField function sends the specified string of characters from the client application to an unprotected field in the current host session.

Prerequisites

You must call ATMConnectSession before using this function.

Syntax

```
rc = ATMSendStringToField ( hWnd, row, column,string)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
row	This integer defines the row position of a field in the connected host session where a string will be sent.
column	This integer defines the column position of a field in the connected host session where a string will be sent.
string	The string to be sent to the host session's unprotected field.

Return Parameters

Result code	
1 or ATM_SUCCESS	Successful.
-1 or ATM_NOTCONNECTED	Not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-5 or ATM_SESSIONLOCKED	Session locked.
-6 or ATM_DATATRUNCATED	Data truncated.
-7 or ATM_INVALIDPOSITION	Invalid presentation space position.
-9 or ATM_SYSTEMERROR	A system error occurred.
-24 or ATM_UNFORMATTEDHOSTPS	Unformatted host presentation space.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.

Comments

This function can be used only in host sessions that have been field-formatted with unprotected fields. This function does not word wrap.

ATMSessionOff

The ATMSessionOff function powers off the currently connected host session. This leaves the terminal window loaded, but shuts off the connection to the host.

Prerequisites

You must call ATMConnectSession before using this function.

Syntax

```
rc = ATMSessionOff (hWnd)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.

Return Parameters

Result Code	
1 or ATM_SUCCESS	Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-9 or ATM_SYSTEMERROR	A system error occurred.
-11 or ATM_RESOURCEUNAVAILABLE	Resource unavailable.
-104 or ATM_NOEMULATORATTACHED	No terminal session is loaded.
-105 or ATM_WSCTRFAILURE	Workstation Control failure.

ATMSessionOn

The ATMSessionOn function powers on the currently connected host session.

Prerequisites

You must call ATMConnectSession before using this function.

Syntax

```
rc = ATMSessionOn (hWnd)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.

Return Parameters

Result Code	
1 or ATM_SUCCESS	Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-9 or ATM_SYSTEMERROR	A system error occurred.
-11 or ATM_RESOURCEUNAVAILABLE	Resource unavailable.
-104 or ATM_NOEMULATORATTACHED	No terminal session is loaded.
-105 or ATM_WSCTRFAILURE	Workstation Control failure.

ATMSetCursorLocation

The ATMSetCursorLocation function places the host cursor at the presentation space location specified by the *row* and *column* parameters.

Note ATMSetCursorLocation is not recommended for use with asynchronous terminals. The host controls cursor positioning, and many host applications do not allow arbitrary cursor positioning.

Prerequisites

You must call ATMConnectSession before using this function.

Syntax

```
rc = ATMSetCursorLocation ( hWnd, row,column)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
row	This integer defines the row position where the cursor will be located.
column	This integer defines the column position where the cursor will be located.

Return parameters

Result Code	
>0	Success: the beginning presentation space position of the string.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-7 or ATM_INVALIDPOSITION	Invalid presentation space position.
-9 or ATM_SYSTEMERROR	A system error occurred.

ATMSetParameter

The ATMSetParameter function sets or changes the current system parameter settings. With the *paramIndex* parameter you can specify the following:

• Whether to convert non-ASCII characters into blanks
• Whether to set focus to the host access software on an ATMConnectSession call
• Whether data passed between a PC and the host should contain the field attribute byte
• How your host access software should address a pause
• Whether your host access software should scan the entire session presentation space during searches or scan from a specific location
• Which direction your host access software searches on the presentation space
• The number of half-second cycles to wait before terminating a host request
• Your host access software's turning off or on of the Windows trace facility
• The manner in which your host access software handles a call to ATMWait
• How data copied between the PC and the host should be translated (ASCII and EBCDIC)
• How the escape character is used with control codes

For a complete list of the system parameters, settings, and corresponding index numbers, see Appendix B, "System Parameter Settings."

Prerequisites

None.

Syntax

```
rc = ATMSetParameter ( hWnd, paramIndex, setting, escape)
```

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application. paramIndex This integer indicates the parameter you want to change. Refer to Appendix B, "System Parameter Settings," for more information.
setting	This integer specifies which paramIndex settings you want to change. Refer to Appendix B, "System Parameter Settings," for more information.
escape	Defines the character to be used instead of the at sign (@) (the default for host AID keys).

Return Parameters

Result code	
>0	The previous setting of the parameter queried Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-9 or ATM_SYSTEMERROR	A system error occurred.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.

ATMShowLastError

The ATMShowLastError displays the most recent error encountered in a message box. The message box title contains the function name, and the message box displays the error description.

Prerequisites

You must call ATMConnectSession before using this function.

Syntax

```
rc = ATMShowLastError ( hWnd)
```

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.

Return Parameters

Result code

1 or ATM_SUCCESS Successful.
-2 or ATM_INVALIDPARAMETER Invalid parameter.
-101 or ATM_MEMORYUNAVAILABLE Memory unavailable.
-104 or ATM_NOEMULATORATTACHED No terminal session is loaded.

ATMStartKeystrokeIntercept

The ATMStartKeystrokeIntercept function allows an HLLWin to filter any keystrokes entered by a terminal operator. (HLLWin is the window or application that is making calls to the DLL.) It is identified by the window handle (hWnd) used in the HLLAPI calls. The intercepted keystrokes can be received through the ATMGetKeystroke function, replaced by other keystrokes with the ATMSendKey function, or used to trigger other processes.

The HLLWin may intercept keystrokes for more than one session at a time, but sending keystrokes on to the intercepted session requires connection to that session, which may not always be available.

If the HLL_INTERCEPTAIDKEY is specified with this function, the intercepted non-AID keys will be sent to the presentation space and only the AID keys will be captured by the HLLWin.

The StopKeystrokeIntercept function or termination of the HLLWin resets the keystroke interception. If you start an intercept, you should stop it before going to something else.

When keystrokes are available, they are read into your application's data area. Each keystroke is represented by a key code. For a list of all key mnemonics, refer to Appendix D, "PC-Host Keyboard Mnemonics."

Prerequisites

None

Syntax

```
rc = ATMStartKeystrokeIntercept ( hWnd,session,Filter)
```

function.

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
session	The session short name.
Filter	Contains one of these values from the include file: HLL_INTERCEPTAIDKEYS or HLL_INTERCEPTALLKEYS.

Return Parameters

Result code	
0	Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameters exist.
-7 or ATM_INVALIDPOSITION	Invalid presentation space position.
-9 or ATM_SYSTEMERROR	A system error occurred.
-11 or ATM_RESOURCEUNAVAILABLE	Resource unavailable.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.
-104 or ATM_NOEMULATORATTACHED	No terminal session is loaded.

ATMStartSession

The ATMStartSession function starts a terminal session, and defines how the session will appear. ATMStartSession can start the host session as follows:

- A normal window
- An icon
- A maximized window
- A hidden window

Prerequisites

ATMConnectSession must be called prior to using this function.

Syntax

```
rc = ATMStartSession ( hWnd, session,visible)
```

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
session	A string containing the short name of the session you want to start.
visible	A string containing one of the following options: N—Start session as normal M—Start session maximized I—Start session as an icon H—Start session as hidden

Return Parameters

Result code	
1 or ATM_SUCCESS	Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-9 or ATM_SYSTEMERROR	A system error occurred.
-11 or ATM_RESOURCEUNAVAILABLE	Resource unavailable.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.
-105 or ATM_WSCTRLFAILURE	Workstation Control failure.
-200 or ATM_NOMATCHINGPSID	No matching Presentation Space ID.

ATMStopKeystrokeIntercept

The ATMStopKeystrokeIntercept function ends the HLLWin's ability to intercept keystrokes for the specified session.

Prerequisites

ATMStartKeystrokeIntercept

Syntax

```
rc = ATMStopKeystrokeIntercept (hWnd, Session)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
Session	The session short name.

Return Pamameters

Result code	
0	Successful
-2 or ATM_INVALIDPARAMETER	Invalid parameters exist.
-9 or ATM_SYSTEMERROR	A system error occurred.

ATMStopSession

The ATMStopSession function terminates the connected terminal session.

Prerequisites

A call to ATMConnectSession must be made prior to using this function.

Syntax

```
rc = ATMStopSession ( hWnd)
```

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.

Return Parameters

Result code	
1 or ATM_SUCCESS	Successful.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-9 or ATM_SYSTEMERROR	A system error occurred.
-11 or ATM_RESOURCEUNAVAILABLE	Resource unavailable.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.
-104 or ATM_NOEMULATORATTACHED	No terminal session is loaded.
-105 or ATM_WSCTRLFAILURE	Workstation Control failure.

Comments

A call to ATMDisconnectSession and ATMResetSystem should be made to complete the terminal closing process.

ATMUnlockKeyboard

The ATMUnlockKeyboard function unlocks the connected host session previously locked by the ATMLockKeyboard function.

Prerequisites

You must call the ATMConnectSession function before using this function.

Syntax

```
rc = ATMUnlockKeyboard ( hWnd)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.

Return Parameters

Result code	
1 or ATM_SUCCESS	Successful.
-1 or ATM_NOTCONNECTED	Not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-9 or ATM_SYSTEMERROR	A system error occurred.

ATMUnregisterClient

The ATMUnregisterClient function disassociates the client's window handle from the EAL. It also does an internal ATMResetSystem.

Prerequisites

ATMRegisterClient

Syntax

```
rc = ATMUnregisterClient ( hWnd)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.

Return Parameters

Result code	
0	This handle has no existing registration.
1 or ATM_SUCCESS	The registration is complete.
-9 or ATM_SYSTEMERROR	A system error occurred.

Comments

Use ATMUnregisterClient to complete your use of the host access. This allows a complete cleanup of the connection. This must be the last EAL function issued in the application.

ATMWait

The ATMWait function waits for the XCLOCK or XSYSTEM messages to be cleared from the OIA of the connected session before returning to the calling application. The OIA can be found near the bottom of your terminal session window.

The 3270 version waits for the XCLOCK to clear. With VT sessions, it waits for the host to idle, simulating XCLOCK. This function is *not recommended* for either 3270 or 5250 because the XCLOCK may flicker on and off. See the **ATMWaitForString, ATMWaitForCursor, and ATMWaitHostQuiet** function descriptions for more reliable Wait methods.

Prerequisites

You must call the ATMConnectSession function before using this function.

Syntax

```
rc = ATMWait ( hWnd)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.

Return Parameters

Result Code	
1 or ATM_SUCCESS	Successful.
-1 or ATM_NOTCONNECTED	Not connected.
-4 or ATM_TIMEOUT	A time-out occurred.
-5 or ATM_SESSIONLOCKED	Session locked.
-9 or ATM_SYSTEMERROR	A system error occurred.
-11 or ATM_RESOURCEUNAVAILABLE	Resource unavailable.
-102 or ATM_DELAYEDBYCLIENT	Delayed by client.

Comments

The OIA contains messages from the host. The host uses this part of the screen when it sends session status information, such as error messages. XCLOCK and XSYSTEM are two such host messages.

The XSYSTEM (System Lock) message means that the host has locked the keyboard after receiving the last request.

By default, ATMWait will time out after one minute, even if the host is still busy. To change this default, use the ATMSetParameter function. The ATMWait parameter determines how your host access software handles a call to ATMWait.

Use the ATMSetParameter function to:

- Produce a time-out after one minute, even if XCLOCK and XSYSTEM are displayed on the OIA.

- Instruct your host access software not to return until the XCLOCK or XSYSTEM messages clear. If this is the setting during a file transfer, ATMWait will not return until the file transfer is complete.

- Instruct your host access software to check the status only and return immediately (with no wait).

For more information about the specific values of the setting of the ATMWait parameter, refer to Appendix B, "System Parameter Settings."

ATMWaitForCursor

This function forces your application to wait until the host cursor has moved to the specified row and column location before continuing.

Prerequisites

ATMConnectSession

Syntax

```
rc = ATMWaitForCursor ( hWnd,row,column, timeOut)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
row	This integer specifies the number of rows from the top of the screen to the cursor location. If row is zero, then ATMWaitForCursor will return when the cursor is in the selected column in any row. Only one of row or column can be zero.
col	This integer specifies the number of columns between the left edge of the screen and the cursor location. If column is zero, then ATMWaitForCursor will return when the cursor is in the selected row in any column. Only one of row or column can be zero.
timeOut	This integer defines the number of half-seconds to wait for the cursor to appear at the row and column location before timing out.

Return Parameters

Result code	
1 or ATM_SUCCESS	Successful.
0 or ATM_NOTFOUND	Unsuccessful.
-1 or ATM_NOTCONNECTED	The session is not connected.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-4 or ATM_TIMEOUT	A time-out occurred.
-5 or ATM_SESSIONLOCKED	Session locked.
-7 or ATM_INVALIDPOSITION	Invalid position.
-9 or ATM_SYSTEMERROR	A system error occurred.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.

ATMWaitForCursorMove

The ATMWaitForCursorMove function waits until the cursor moves and returns the new presentation space position of the cursor.

Prerequisites

An ATMConnectSession call must be made before this function is called.

Syntax

```
rc = ATMWaitForCursorMove ( hWnd,timeout)
```

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
timeout	This integer defines the amount of time to wait for a movement of the cursor, in half-second units.

Return Parameters

Result code	
> 0 or Success	The cursor moved to new presentation space position.
-1 or ATM_NOTCONNECTED	An ATMConnectSession call has not been made for this session.
-4 or ATM_TIMEOUT	The cursor did not move before the time-out interval passed.
-9 or ATM_SYSTEMERROR	A system error occurred.

Comments

Take care when using the returned position. This function returns on the first move of the cursor. The host or user can continue to move the cursor after the position is returned.

ATMWaitForEvent

The ATMWaitForEvent function waits for one of the events that has added to the specified table to occur or until a specified time-out period reached. Use this function after building a table with the functions wait to the table (for example, ATMAddWaitForString, ATMAddWait-ForCursor) to allow any one of a number of conditions to be waited once.

Prerequisites

You must call ATMConnectSession before using this function.

Syntax

```
rc = ATMWaitForEvent ( hWnd,table,timeout)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
table	This integer indicates the table of events to be waited
timeout	This integer defines the amount of time to wait for a movement of the cursor, in half-second units.

Return Parameters

Result code	
> 0 or Success	The event ID was supplied when the event was successfully added.
-1 or ATM_NOTCONNECTED	An ATMConnectSession call has not been made for this session.
-4 or ATM_TIMEOUT	The cursor did not move before the time-out interval passed.
-9 or ATM_SYSTEMERROR	A system error occurred.
-303 or ATM_TABLEMAXEXCEEDED	The table ID is greater than the maximum allowed.
-304 or ATM_TABLENOTSET	No wait entries were added to the table.

Comments

This function should be used after sending a string or key to the host one of the events to occur. ATMWaitForEvent establishes a watch the events specified in the table and for the time-out, and then resumes host data display if an ATMHoldHost has been issued. When the function returns, the hold host is reestablished, if present.

When you are working in an asynchronous terminal environment, ATMHoldHost may be needed before *both* the ATMWaitForEvent last sending of the string or key that is expected to cause one of the events.

ATMWaitForHostConnect

The ATMWaitForHostConnect function waits until a host connection is established. It returns immediately if the host access software is already connected to a host.

Prerequisites

An ATMConnectSession call must be made before this function is called.

Syntax

```
rc = ATMWaitForHostConnect (hWnd,timeout)
```

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
timeout	This integer defines the amount of time to wait for a host connection to occur, in half-second units.

Return Parameters

Result code	
1 or ATM_SUCCESS	The host access software disconnected from the host.
-1 or ATM_NOTCONNECTED	An ATMConnectSession call has not been made for this session.
-4 or ATM_TIMEOUT	A host disconnection did not occur before the time-out interval passed.
-9 or ATM_SYSTEMERROR	A system error occurred.

Comments

Note that this is a *host* connection. The term “connection” here is completely separate from the term “connection” used in the ATMConnectSession function.

ATMWaitForHostDisconnect

The ATMWaitForHostDisconnect function waits until a host connection is broken. It immediately returns if the host access software is already disconnected from the host.

Prerequisites

An ATMConnectSession call must be made before this function is called.

Syntax

```
rc = ATMWaitForHostDisconnect ( hWnd,timeout)
```

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
timeOut	This integer defines the amount of time to wait for a host disconnection to occur, in half-second units.

Return Parameters

Result code	
1 or ATM_SUCCESS	The host has been disconnected from the host access software.
-1 or ATM_NOTCONNECTED	An ATMConnectSession call has not been made for this session.
-4 or ATM_TIMEOUT	A host disconnection did not occur before the time-out interval passed.
-9 or ATM_SYSTEMERROR	A system error occurred.

Comments

This is a *host* connection. This “connection” is completely separate from the “connection” used in the ATMConnectSession function. Two kinds of connections exist—from the client application to the host access software (a *HLLAPI* connection) and from the host access software to the host (a *host* connection).

ATMWaitForKey

The ATMWaitForKey function waits until the given keystroke is entered on the keyboard. It does not respond to a key sent with ATMSendKey.

Prerequisites

You must call ATMConnectSession before using this function.

Syntax

```
rc = ATMWaitForKey (hWnd,key,timeout)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
key	This string contains the key to be waited for (see Appendix D, "PC-Host Keyboard Mnemonics," for ATMSendKey codes).
TimeOut	This integer defines the time to wait for a specified keystroke to be entered, in half-second units.

Return Parameters

Result code	
1 or ATM_SUCCESS	The host has been disconnected from the host access software.
-1 or ATM_NOTCONNECTED	An ATMConnectSession call has not been made for this session.
-4 or ATM_TIMEOUT	A host disconnection did not occur before the time-out interval passed.
-9 or ATM_SYSTEMERROR	A system error occurred.

Comments

The key can be specified as the ASCII code of the key or use the same escape character codes as ATMSendKey (@E for ENTER, @D for DELETE, and so on).

ATMWaitForString

The ATMWaitForString function waits until a specified string of text appears in the host session, or until a specified time-out period is reached. Use this function after the ATMSendKey function to make sure the host application has updated the screen before your application attempts to send more keystrokes. This function helps avoid timing problems.

Prerequisites

You must call ATMConnectSession before using this function.

Syntax

```
rc = ATMWaitForString (hWnd, row,column,waitString,timeout)
```

Call Parameters

Parameter	Description
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
row	This integer defines the row position of a field in the connected host session where waitString will be found. If row is set to zero, all rows will be searched.
column	This integer defines the column position of a field in the connected host session where waitString will be found. If column is set to zero, all columns will be searched.
waitString	This string contains the text you want to locate.
timeout	This integer defines the number of half-seconds to search before timing out.

Return Parameters

Result Code	
>0	The beginning presentation space position of the string: success.
0	A time-out occurred before the string was found.
-1 or ATM_NOTCONNECTED	An ATMConnectSession call has not been made for this session.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-4 or ATM_TIMEOUT	A time-out occurred.
-5 or ATM_SESSIONLOCKED	Session locked.
-7 or ATM_INVALIDPOSITION	Invalid presentation space position.
-9 or ATM_SYSTEMERROR	A system error occurred.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.

Comments

If *row* equals zero, all rows will be searched at the column specified by *col*. If *col* equals zero, all columns will be searched at the row specified by *row*. If both *row* and *col* are zero, the entire presentation space will be searched.
If the string is found before the time period specified by the *timeout* parameter has elapsed, the function will return immediately. Otherwise, it continues searching until time runs out.

ATMWaitHostQuiet

This function waits until the XCLOCK has disappeared from the Operator Information Area for the specified number of milliseconds. Because the host XCLOCK can flash off and on several times during processing, a *settleTime* parameter is provided. This parameter may need to be optimized for your host system. When ATMWaitHostQuiet is used with VT, host idle is used instead of XCLOCK.

Prerequisites

ATMConnectSession

Syntax

```
rc = ATMWaitHostQuiet ( hWnd,settleTime, timeOut)
```

Call Parameters

Parameter	**Description**
rc	The integer return value.
hWnd	This integer indicates the window handle of your application.
settleTime	This integer defines the number of milliseconds the XCLOCK must be off before returning to calling application.
timeOut	This integer defines the maximum number of half seconds the function should wait while the XCLOCK is still on before returning.

Return Parameters

Result code	
1 or ATM_SUCCESS	Successful.
-1 or ATM_NOTCONNECTED	An ATMConnectSession call has not been made for this session.
-2 or ATM_INVALIDPARAMETER	Invalid parameter.
-4 or ATM_TIMEOUT	A time-out occurred.
-5 or ATM_SESSIONLOCKED	Session locked.
-9 or ATM_SYSTEMERROR	A system error occurred.
-101 or ATM_MEMORYUNAVAILABLE	Memory unavailable.

Appendix A: General troubleshooting procedures

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

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

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

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

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

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

Appendix B: Host keyboard mnemonics

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

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

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

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

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

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

Symbols used throughout the tables have the following meanings:

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

Table B-1. Windows keyboard mnemonics

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

Appendix C: System Parameter Settings

ParamIndex and Settings parameter values

This appendix describes global parameters available with the ATMSetParameter, ATMGetParameter, and ATMReset functions.

ParamIndex specifies which of the eleven separate global parameters (index 5 is not used) will be affected, and *Settings* defines the specific setting for the global parameters. To examine the global parameters, call the ATMGetParameter function.

ATMSetParameter

The ATMSetParameter function sets a specified global parameter for the client window, plus the Escape global parameter. Values set in this function affect other functions. Syntax for the ATMSetParameter call is as follows:

```
ATMSetParameter (hWnd,ParamIndex,Setting,Escape)
```

ATMGetParameter

To examine the global parameters, call the ATMGetParameter function.

ATMResetSystem

ATMResetSystem reinitializes the system parameters to their default values and disconnects any previously connected sessions. Default values are listed in the parameter descriptions which follow.

Parameter Index	Purpose	Setting	Description
ATM_ATTRIB (1)	Specifies whether to convert non-ASCII characters to blanks	1	Disables conversion
		2	Default. Enables conversion
ATM_AUTORESET (2)	Tries to reset all KB inhibited conditions by sending RESET prefixed to all strings sent via ATMSendKey	1	Default. Enables RESET
		2	Disables RESET attempt
ATM_CONNECTTYPE (3)	Specifies whether to set application focus to host access software at ATMConnectSession	1	Default. Performs logocal connection
		2	Performs physical connection by bringing host access software to top of desktop Z-axis and giving it focus.

Parameter Index	Purpose	Setting	Description
ATM_EAB (4)	Specifies whether data passed between PC and host contains EAB	1	Enables transmission of EABs, yielding two characters for each session character.
		2	Default. passes session data without EABs.
ATM_SEARCHORG (6)	Specifies whether to scan entire presentation space during searches	1	Default. Scans the entire PS of the session.
		2	Scans only from specified starting point on searches.
ATM_SEARCHDIRECTION(7)	Specifies which direction to scan the presentation space.	1	Default. Searches from top left corner to bottom right
		2	Searches from bottom right to top left
ATM_TIMEOUT(8)	Specifies the number of 30-second cycles to wait before terminating a file transfer request.	0	Default. Allows file transfer to run to completion
ATM_TRACE(9)	Specifies whether to trace API calls in the host access software trace window. (Supports async trace reporting only, and only if trace facility window is open.)	1	Reports all HLLAPI calls in the trace facility's window.
		2	Default. Does not report HLLAPI calls.
ATM_WAIT(10)	Specifies how host access software is to handle calls to ATMWait.	1	Default. Times out a wait for XCLOCK or XSYSTEM to clear after one minute.
		2	Returns from an ATMWait call after XCLOCK or XSYSTEM has cleared.
		3	Checks XCLOCK / XSYSTEM status and returns immediately
ATM_XLATE(11)	Specifies whether to translate application data to and/or from EBCDIC in session presentation space. 3270 only, PS data for 5250 and async is in ASCII already.	1	Does not convert between ASCII and EBCDIC when copying application data to and/or from presentation space
		2	Default. Converts 3270 PS data to/from ASCII as needed
ATM_ESCAPE(12)	ASCII value of char preceding keyboard mnemonic escape sequence. where, for example '@T' is the mnemonic for 'Tab'.	n	Default value is asc('@'). Can be any value except 0x20, because blank is not a valid escape character.

Appendix D: Extended Attributes

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

3270 Character Attributes

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

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

5250 Character Attributes

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

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

Color Attributes

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

Bit	**Meaning**
0–3	Background character color 0000 = Black 0001 = Blue 0010 = Green 0011 = Cyan 0100 = Red 0101 = Magenta 0110 = Brown (3270), Yellow (5250) 0111 = White
4-7	Foreground character color 0000 = Black 0001 = Blue 0010 = Green 0011 = Cyan 0100 = Red 0101 = Magenta 0110 = Brown (3270), Yellow (5250) 0111 = White 1000 = Gray 1001 = Light blue 1010 = Light green 1011 = Light cyan 1100 = Light red 1101 = Light magenta 1110 = Yellow 1111 = White (high intensity)

Appendix D

This appendix outlines the syntax and criteria for using the ATMSendFile and ATMReceiveFile functions for sending or receiving data files between the mainframe and PC in the VM/CMS, MVS/TSO, and VT supported environments.

The *Command* parameter contains the required file-transfer-option parameters.

ATMSendFile transfers a file from your PC to the host, while the ATMReceiveFile function transfers a file from the host to the PC.

The syntax for ATMReceiveFile is as follows:

```
      rc = ATMReceiveFile(hWnd,Command,CommandLength)
```

The syntax for ATMSendFile is as follows:

```
      rc = ATMSendFile(hWnd,Command,CommandLength)
```

File transfers with VM/CMS

In a VM/CMS session, the syntax for the ATMSendFile and ATMReceiveFile *Command* parameter is as follows:

```
[d:][path]filename[.ext] w: fn [ft] [fm] [(options]
```

The syntax conventions are as follows:

• Brackets ([ ]) indicate that the enclosed parameter is optional. To include an optional parameter in a function call, omit the brackets.

• Type periods, colons, and parentheses where indicated. When they appear in brackets, use them only if you are including the accompanying options.

• In the command string, an open left parenthesis indicates that one or more options follow.

• ASCII is a reserved word and can only be used in the optional parameters. For example, the mainframe filename cannot contain the string “ASCII.”

See the following section, “VM/CMS file-transfer parameters,” for a complete list of command parameters.

Note To transfer files in CMS, you must be in CMS command mode (indicated by a Ready prompt) or at a command line prompt in a PROFS session. If you have any questions about CMS and file transfers, contact your system administrator.

VM/CMS file-transfer parameters

Parameter	Definition
[d:]	The drive to which the PC file is sent or received. If no drive is specified, the file transfer defaults to the current drive.
[path]	The directory path of the PC file to be sent or target file to be received. If no path is specified, path defaults to the current directory on the PC.
filename	The PC file to be sent or received.
[. ext]	The extension of the PC filename to be sent or received.

w:	The target session's short name. This must be the session in which you have established a CMS session for sending or receiving files. If w is not specified, the first open mainframe session is the target.
fn	The name of the file being sent to or received from the mainframe.
[ft]	The file type of the mainframe file to be sent or received, if any.
[fm]	An optional parameter indicating the file mode. The parameter indicates to CMS which disk drive the file is sent to (for SendFile only). This parameter defaults to the A drive. (options One or more of the options described in Table C-2. The option list must be preceded by a single left parenthesis.

VM/CMS file-transfer options

Option	**Definition**
append	Adds the transferred file to the end of an existing file, overriding any specified values for the lrecl and recfm options. **Note** The existing file is replaced if append is not used and the filename already exists on the mainframe and PC.
ascii	Converts ASCII to EBCDIC when a file is sent to the host, and converts EBCDIC to ASCII when the PC receives a file from the host. ASCII mode is recommended when transferring text files.

bufsize	Enables HLLAPI to select the buffered file-transfer option. The default for a DFT session is a structured-field file transfer. If -B appears in the command string, HLLAPI will invoke a buffered file-transfer instead. (CUT sessions support only buffered file transfers.) The command string for a structured-field file transfer may set the size of structured field buffers, in kilobytes, with one of the following:
	BUFSIZE = 2 (default)
	BUFSIZE = 4
	BUFSIZE = 8
	BUFSIZE = 32

crlf	Deletes carriage-return and line-feed characters when sending files to the host, and appends them when receiving files from the host. Usually, PC file alphanumeric data contains a crlf character at the end of each line to denote the end of each record. Deleting these characters using the crlf option lets you read the file more easily on the host.
lrecl n	(ATMSendFile only.) Specifies that the mainframe file will contain fixed-length records of size n. Records are padded with space characters if they are shorter than lrecl or the default length.
rrecfm f	(ATMSendFile only.) Specifies that the file will contain fixed length records.
recfm v	(ATMSendFile only.) Specifies that the file will contain variable length records. **Note** If you use the append option, the recfm option is ignored. If you do not specify recfm, the default is used (new files) unless you specify crlf. When you specify crlf, the default is v (variable length). For existing files, the default is the recfm of the existing file. We recommend that you use the recfm v option without the crlf option when mainframe record padding or deletion of trailing spaces must not occur.

File transfers with MVS/TSO

The syntax for the *lpszCommand* parameter for the ATMSendFile and ATMReceiveFile functions in a MVS/TSO session is as follows:

```
[d:][path]filename[.ext] w: dsn [(member)] [lpw] [options]
```

Note that you should enter periods, colons, and parentheses where indicated. When they appear in brackets, enter them only if you are using the accompanying options.

See the following section, "MVS/TSO file-transfer parameters," for a complete list of command parameters.

Note To transfer files in TSO, you must be in TSO command mode, usually indicated by a Ready prompt. If you have any questions about TSO and file transfers, consult your system administrator.

MVS/TSO file-transfer parameters

Parameter Definition	
[d:]	This parameter defines the drive where the PC file is to be sent or received. If no drive is specified, the file transfer defaults to the current drive.
[path]	The directory path of the PC file to be sent or the target file to be received. If no path is specified, path defaults to the current PC directory.
filename	The PC file to be sent or received.
[.ext]	The extension of the PC file to be sent or received.
w:	The target session's short name. This must be the session in which you have established a TSO session for sending or receiving files. If w is not specified, the first open mainframe session is the target.
dsn	The TSO dataset name you assign to the file when you send it to the mainframe, or the TSO dataset name of the file to be received.
[(member)]	An optional parameter that tells TSO to save or receive the file as a PDS to save mainframe disk space.

MVS/TSO file-transfer options

Option	**Definition**
append	Adds the transferred file to the end of an existing file. The append option overrides any specified values for the lrecl and recfm options. **Note** If append is not used and the filename already exists on the mainframe and PC, the existing file is replaced. Append is not valid with a PDS.
ascii	Converts EBCDIC to ASCII when a file is sent to the host, and converts ASCII to EBCDIC when the PC receives a file from the host. ASCII mode is recommended when transferring text files.
crlf	Deletes carriage-return and line-feed characters when sending files to the host, and appends them when receiving files from the host. Usually, alphanumeric data in a PC file contains a crlf character at the end of each line to denote the end of each record. Deleting these characters using the crlf option lets you read the file more easily on the host.

lrecl(n)	ATMSendFile only. Specifies the logical record length of the mainframe file, where n is the number of characters in each record. The default is 80 for new files. If you are replacing a file, the default is the lrecl of the existing file. If you have specified the append option, lrecl is ignored. When working with variablelength records, n is the maximum length record the mainframe will accept.
blksize(n)	ATMSendFile only. Specifies the block size of the mainframe dataset, where n is the length in bytes of a data block. If you omit blksize, the default is lrecl + 4 for new files. If you are appending or replacing a dataset, blksize is ignored.
recfm(f)	ATMSendFile only. Specifies that the mainframe file will contain fixed-length records of length f. Records are padded with space characters if they are shorter than lrecl or the default length.
recfm(u)	ATMSendFile only. Specifies that the mainframe file will contain records of undetermined length.
recfm(v)	ATMSendFile only. Specifies that the file will contain variable length records. **Note** If you use the append option, the recfm option is ignored. If you do not specify recfm, the default is used (for new files) unless you specify crlf. When you specify crlf, the default is v (variable length). For existing files, the default is the recfm of the existing file. We recommend that you use the recfm(v) option without the crlf option when mainframe record padding or deletion of trailing spaces must not occur.
-B	Use this option only if it is unnecessary to use structured fields.
-S	Indicates that you want to perform a structured-field file transfer. This option is only valid in the DFT environment. space ATMSendFile only. Specifies the amount of space you want to allocate for a new dataset. If you use space, the following syntax applies: space(quantity[, increment]) [avblock(value) tracks cylinders] The three options for increment, avblock(value), tracks, or cylinders, specify the number of units of the initial quantity and expansion increment size. If you do not specify any of these options, the default is blksize.

Using the WaitForEvent Functions

What is a WaitForEvent function?

EAL supplies a number of functions that allow an application to wait for a single specific condition on the emulated screen. WaitForEvent functions include ATMWait, ATMWaitForCursor, ATMWaitForKey, and ATMWaitForString, among others.

A host can respond in a number of ways. There may be a variety of error conditions, network broadcast messages, or, in the case of asynchronous scrolling applications, the conditions may not occur in exactly the same way each time. The ATMWaitForEvent functions and associated table maintenance functions make it possible to wait for more than one condition with one call and then respond to the condition. (Counting a time-out, some functions allow for two conditions.) These functions also work together to build and maintain one or more tables of events. The ATMWaitForEvent function then waits until one of the events is triggered or a time-out occurs.

The following is a list of all functions covered by this appendix:
- ATMAddWait
- ATMAddWaitForCursor
- ATMAddWaitForCursorMove
- ATMWaitForEvent
- ATMAddWaitForHostDisconnect
- ATMAddWaitForKey
- ATMAddWaitForString
- ATMAddWaitForStringNotAt
- ATMAddWaitHostQuiet
- ATMClearEventTable
- ATMDeleteEvent
- ATMHoldHost
- ATMResumeHost

Tables of events

The ATMWaitForEvent function works by waiting for one of a set of events (called a table) to occur. The ATMWaitForEvent function waits for the first occurrence of an event in the supplied table identifier and then returns a value indicating which event has been triggered.

Before you can use ATMWaitForEvent you must build the table. This is done by using other functions to add events to a table.
The following is the general form of these Add functions:

```
ATMAddnameofevent(handle,TableID,EventID,other event parameters)
```

Here’s an example:
`ATMAddWaitForString(handle,2,12,5,25,”UserName:”)`

This adds an event to table 2 with an event identifier of 12. The event waits for the string “UserName:” to appear in row 5, column 25.
In general, the name of the add function is the same as the old singleton wait function with `Add` placed right after `ATM`. The first three parameters supplied are the session identifier, the table being added to, and the value that will be returned if this specific event triggers first. Other event parameters are identical to those supplied to the singleton function, but there is no time-out parameter supplied at the end of the wait functions. (This is supplied as the last parameter to ATMWaitForEvent.)

Once the event table has been built, the ATMWaitForEvent function may be called. The return code will be the event identifier of the event which occurs first, or -4 for a time-out.

Maintaining a table of events

It is not possible to change the event for a particular event ID. To reuse an event ID value, the event must be deleted with the ATMDeleteEvent function and then a new event added with the same event ID.
The other table maintenance function is ATMClearEventTable. It works like the ATMDeleteEvent function, deleting all the events that have been added to a table. If you attempt to delete an event from a table or clear a table with no events, an error message will be returned. An error message will also be returned if you try to wait for an event on an empty table.

Multiple tables

By using the table ID you can maintain and use multiple tables. However, ATMWaitForEvent waits only for the table specified as the second parameter. Currently, there is a limit to the number of tables you can use and the supplied Table ID cannot be an integer higher than this number.

Tables per session ID

The tables that EAL maintains are kept separately for each session ID (window handle) you supply. You can use a table, disconnect, and come back later to reuse it. The table remains intact until the session is shut down, you clear the table, or you issue ATMResetSystem or ATMUnregisterClient. The limit to the number of tables is per session ID.

Performance

Allowing for a larger number of tables will decrease performance slightly.

Glossary

Configured Sessions (IsConfigured)—This statement indicates whether the 3270 async configured session exists—sitting on the desktop with a configuration file (.EDP) or in a database (.KTC file).

Connected (IsConnected)—This statement indicates whether the session is connected (no correspondance) with a HLLAPI connection to it.

Emulated Powered—This statement indicates that the emulated session has no correspondance and the emulated screen is visible (for KEA95). KEA! for 16-bit async does not distinguish between opened and powered; if it is is opened it is powered and vice versa.

Emulated Sessions (IsEmulated)—The host access software owns the session.

Opened Sessions (IsOpened)—This statement indicates that the opened configuration file in the database has been opened and a session is running.

Powered Sessions (IsPowered)—This statement indicates that the session is powered and turned on (connection to host or VTAM) and running a session able to talk to the host.