Converting Attachmate's EAL to NetManage® HLLAPI

Product: RUMBA® Tools products
Version #: N.A.
Host: N.A.
NIC: N.A.
Interface: HLLAPI
Oper Sys: Microsoft® Windows®

Summary

Attachmate® provides a DLL interface to its Enterprise Access Library (EAL). NetManage provides a similar capability via the RUMBA Developer Library (RDL). This technical bulletin outlines a subset of Attachmate's HLLAPI and EAL functions and shows the equivalent RUMBA Developer Library functions or individual HLLAPI calls.

Comparing HLLAPI Functions

The RUMBA Developer Library (RDL) does not require a DLL but contains functions that can be used within your development environment. These functions are available today in Visual Basic® and PowerBuilder® formats, but could easily be used in any other development environment. RUMBA Tools for Visual Basic, RUMBA Tools for PowerBuilder, and RUMBA Tools for HLLAPI (C/C++) all document the capabilities of the RDL and the NetManage implementation of HLLAPI. NetManage has made every attempt to follow the IBM® specification for HLLAPI implementation and return codes.

Some of the functions noted can be converted into individual HLLAPI functions; others will be RDL functions (a superset of HLLAPI functions).

Attachmate's HLLAPI-based EAL and other functions in ATMAPI.DLL	NetManage's HLLAPI-based RDL functions and other functions in EEHLLAPI.DLL	Extended NetManage description
ATMConnectSession (ByVal *hWnd* As Integer, ByVal *Session* As String) As Integer Specifies the host session with which to interact Possible return codes: 1, -2, -5, -9, -200	WD_ConnectPS (ByVal *hInstance* As Integer, ByVal *ShortName* As String) As Integer *OR* RDL_RunConnect(Session As String, cPSId As String, hApp As Integer, ScrnId As String, Row As Integer, Col As Integer, WaitSeconds As Integer, NumPSUpdates As Integer, WindowState As Integer) As Integer	Establishes a connection to a RUMBA session; i.e., presentation space; operation affected by SetSessionParameters() Possible return codes: 0, 1, 5 (successful but inhibited), 9, 11 RDL routine: RunConnect. This performs the same function as ATM. Run a predefined profile and connect your EHL application to this profile
ATMDisconnectSession (ByVal *hWnd*	WD_DisconnectPS (ByVal *hInstance*	Releases a "pre-connected" RUMBA

As Integer) As Integer Releases the previously connected host session Possible return codes: 1, -1, -2, -9, -200	As Integer) As Integer *OR* RDL_CloseDisconnect(cPSId As String, hApp As Integer, CloseFlag As Integer) As Integer	session from use. Possible return codes: 0, 1, 9 RDL routine: CloseDisconnect performs the same function as ATM. However, you have the choice of deleting the specific session or simply disconnecting the HLLAPI application.
ATMGetConnectionStatus (ByVal *hWnd* As Integer, ByVal *StatusType* As Integer) As Integer Returns status codes describing the current connection; StatusType = 1 is XSTATUS, 2 is ATM_CONNECTION, 3 is ATM_ERROR Possible return codes: see table below	RDL_CheckSessionStatus(hApp As Integer, cPSId As String) As Integer EHL Status: QuerySessions or QuerySessionStatus - both give information about the current sessions available for EHLLAPI application	Tries to resolve errors when encountered in the OIA.
ATMGetCursorLocation (ByVal *hWnd* As Integer) As Integer Returns the cursor location as a presentation space position Possible return codes: location, -1, -2, -9	WD_QueryCursorLocation (ByVal *hInstance* As Integer, *Location* As Integer) As Integer	Provides the location of the cursor in the presentation space; operation affected by SetSessionParameters() Possible return codes: 0, 1, 9
ATMGetError (ByVal *hWnd* as Integer, ByVal *ErrString* As String, ByVal *Length* As Integer) As Integer Provides the name of function in which most recent error occurred and provide description of error; these two items are separated by a new-line character in ErrString Possible return codes: number of characters in ErrString, 0 (no errors), -2, -101, -104	T4VB: The RUMBAError event will be triggered when an EHLLAPI function receives an error. RDL/EHL: If using the RDL or EHL functions, there isn't a NetManage equivalent available to-date. You would have to program each function call and return as global values that you could access.	>> Request enhancement.
**ATMGetFieldLength (ByVal *hWnd* As Integer, ByVal *Row* As Integer, ByVal *Column* As Integer, ByVal *PosType* As Integer) As Integer Returns the length of a field given a row, column, and relative position of field; PosType = 0 is ATM_THISFIELD, 1 is ATM_NEXTFIELD, 2 is ATM_PREVIOUSFIELD, 3 is ATM_NEXTPROTECTEDFIELD, 4 is ATM_NEXTUNPROTECTEDFIELD, 5 is ATM_PREVIOUSPROTECTEDFIELD, 6 is ATM_PREVIOUSUNPROTECTEDFIELD Possible return codes: field length, 0 (field not located), -1, -2, -7, -9 , -101	WD_FindFieldLength (ByVal *hInstance* As Integer, *Length* As Integer, ByVal *Position* As Integer, ByVal *FindData* As String) As Integer T4VB: Test tool provided: FieldSpy	Provides the length of a field given a presentation space position and relative field position (FindData = " " (two blank) for current field, "T " (T blank) for current field, "P " (P blank) for previous field, "N " for next field, "NP" for next protected, "NU" for next unprotected, "PP" for previous protected, "PU" for previous unprotected) Possible return codes: 0, 1, 2, 7, 9, 24, 28
**ATMGetFieldPosition (ByVal *hWnd* As Integer, ByVal *Row* As Integer, ByVal *Column* As Integer, ByVal	WD_FindFieldPosition (ByVal *hInstance* As Integer, *Location* As Integer, ByVal *Position* As Integer,	Provides the location (as a presentation space position) of a field identified by its position and relative field position

PosType As Integer) As Integer Returns the presentation space position of a field identified by row, column, and relative position of field; PosType = 0 is ATM_THISFIELD, 1 is ATM_NEXTFIELD, 2 is ATM_PREVIOUSFIELD, 3 is ATM_NEXTPROTECTEDFIELD, 4 is ATM_NEXTUNPROTECTEDFIELD, 5 is ATM_PREVIOUSPROTECTEDFIELD, 6 is ATM_PREVIOUSUNPROTECTEDFIELD Possible return codes: field position, 0 (field not located), -1, -2, -7, -9, -101	ByVal *FindData* As String) As Integer T4VB: Test tool provided: FieldSpy	(FindData = " " (two blank) for current field, "T " (T blank) for current field, "P " (P blank) for previous field, "N " for next field, "NP" for next protected, "NU" for next unprotected, "PP" for previous protected, "PU" for previous unprotected) Possible return codes: 0, 1, 2, 7, 9, 24, 28
ATMGetSessions (ByVal *hWnd* As Integer, ByVal *SessionList* As String, ByVal *Length* As Integer, ByVal *State* As Integer) As Integer Returns the number of or provides, in SessionList, names of sessions matching certain state criteria; State = 1 is ATM_GETCONFIGURED, 2 is ATM_GETOPENED, 3 is ATM_GETPOWERED, 4 is ATM_GETCONFIGUREDCOUNT, 5 is ATM_GETOPENEDCOUNT, 6 is ATM_GETPOWEREDCOUNT, 11 is ATM_GETEMULATED, 12 is ATM_GETEMULATEDPOWERED, 14 is ATM_GETEMULATEDCOUNT, 15 is ATM_GETEMULATEDPOWEREDCOUNT Possible return codes: number of characters copied to SessionList or count of sessions meeting state criteria, 0 (no session found), -2, -9, -101	WD_QuerySessions (ByVal *hInstance* As Integer, *Count* As Integer, *QuerySessionData* As QUERYSESSIONSTRUCT) As Integer {Incomplete compatibility with the state criteria not being available}	Provides a count of active sessions and, in QuerySessionData, the short name of each session with their respective presentation sizes Possible return codes: 0, 2, 9
ATMGetSessionStatus (ByVal *hWnd* As Integer, ByVal *Session* As String, ByVal *StatusType* As Integer) As Integer Provides a session condition based on status; StatusType = 1 is ATM_ISCONFIGURED, 2 is ATM_ISOPENED, 3 is ATM_ISPOWERED, 14 is ATM_ISEMULATED, 15 is ATM_ISCONNECTED, 16 is ATM_ISFILETRANSFER Possible return codes: 1, 0, -1, -2, -9, -101, -200	QuerySessions would list the RUMBA sessions available for the EHL app to connect to. An rc=1 from ConnectPS would indicate that the session has not connected to the host. Once ConnectPS succeeds, use RDL_CheckSessionStatus (or CopyOIA to discern status of host)	
ATMGetString (ByVal *hWnd* As Integer, ByVal *Row* As Integer, ByVal *Column* As Integer, ByVal *GetString* As String, ByVal *Length* As Integer) As	RDL_GetPSArea (*hApp* As Integer, *cPSId* As String, *row* As Integer, *col* As Integer, *Length* As Integer, *HostValue* As String) As Integer	Copies characters from the presentation space delimited by row, column, and length (hApp is the instance handle and cPSId is the EHLLAPI short name of

Integer Copies a string from the connected session given row, column, and length; Length is strictly the number of characters to copy Possible return codes: 1, -1, -2, -4, -5, -7, -9	*OR* WD_CopyPSToString(ByVal hInstance As Integer, ByVal Position As Integer, ByVal Buffer As String, ByVal Length As Integer) As Integer *OR* WD_CopyBlockToString(ByVal hInstance As Integer, ByVal StartPosition As Integer, ByVal EndPosition As Integer, ByVal Buffer As String) As Integer	the RUMBA session) Possible return calls: presentation space position, 0, 1, 2, 4, 5, 7, 9, 26, 9998, 9999
**ATMGetStringFromField (ByVal *hWnd* As Integer, ByVal *Row* As Integer, ByVal *Column* As Integer, ByVal *GetString* As String, ByVal *Length* As Integer) As Integer Copies a string from a host field specified by row, column, and length; Length is strictly the number of characters to copy Possible return codes: 1, -1, -2, -5, -6, -7, -9, -24, -101	WD_CopyFieldToString (ByVal *hInstance* As Integer, ByVal *Position* As Integer, ByVal *Buffer* As String, ByVal *Length* As Integer) As Integer	Copies characters from a field in the presentation space; operation affected by SetSessionParameters() Possible return codes: number of bytes in source field or calling data string, 0, 1, 2, 6, 7, 9, 24
ATMResetSystem (ByVal *hWnd* As Integer) As Integer Re-initializes all system parameters and disconnects all sessions Possible return codes: 1, -2, -9	WD_ResetSystem (ByVal *hInstance* As Integer) As Integer	Re-initializes EEHHLAPI.DLL: reset session parameters to defaults, release presentation spaces, disconnect session, stop keystroke interception and host update monitoring Possible return codes: 0, 9
ATMRowColumn (ByVal *hWnd* As Integer, ByVal *Position* As Integer, ByVal *Toggle* As Integer) As Integer Converts a presentation space position to a row or column position, one at a time, depending on the Toggle option Possible return codes: column or row position, -1, -2, -7, -9, -101, -103	WD_Convert (ByVal *hInstance* As Integer, ByVal *ConvertType* As Integer, *RowCol* As ROWCOL, ByVal *ShortName* As String) As Integer	Converts a presentation space position to row/column coordinates or vice-versa (ConvertType of 'P' or CONVERT_POSTITION to convert from position, ConvertType of 'R' or CONVERT_ROW to convert from row/column) Possible return codes: presentation space position or column number, 9 (must check for system error since this may be position/column number), 9998, 9999
**ATMSendAndWait (ByVal *hWnd* As Integer, ByVal *Row* As Integer, ByVal *Column* As Integer, ByVal *SendString* As String, ByVal *WaitString* As String, ByVal *TimeOut* As Integer) As Integer Sends a keystroke represented by ASCII string and searches, with timeout as a number of half-seconds, the next host screen for a particular string Possible return codes: 1, 0 (not found), -1, -2, -5, -7, -9, -101	RDL_ChangeHostScreen(cPSId As String, hApp As Integer, cKey As String, cUniqueId As String, Row As Integer, Col As Integer, WaitTime As Integer, NumPSUpdates As Integer) As Integer	Provides a host verification technique that counts host screen updates.
ATMSendKey (ByVal *hWnd* As Integer, ByVal *Key* As String) As Integer Sends keystrokes represented by ASCII	WD_SendKey (ByVal *hInstance* As Integer, ByVal *Buffer* As String) As Integer	Sends a string of characters and/or ASCII mnemonics to the current session; operation affected by

string Possible return codes: 1, -1, -2, -4, -5, -9	Use SendKey for sending mnemonics, but use CopyStringtoPS if you want faster response from sending strings.	SetSessionParameters Possible return codes: 0, 1, 2, 4, 5, 9
ATMSendString (ByVal *hWnd* As Integer, ByVal *Row* As Integer, ByVal *Column* As Integer, ByVal *SendString* As String) As Integer Sends a string to the host session at a specified row, column Possible return codes: 1, -1, -2, -5, -7, -9	WD_CopyStringToPS (ByVal *hInstance* As Integer, ByVal *Position* As Integer, ByVal *Buffer* As String, ByVal *Length* As Integer) As Integer	Copies an ASCII string to the current session at specified position; operation affected by SetSessionParameters() Possible return codes: 0, 1, 2, 5, 6, 7, 8, 9
**ATMSendStringToField (ByVal *hWnd* As Integer, ByVal *Row* As Integer, ByVal *Column* As Integer, ByVal *SendString* As String) As Integer Sends a string to an unprotected field in the host session at a specified row, column Possible return codes: 1, -1, -2, -5, -6, -7, -9, -24, -101	WD_CopyStringToField (ByVal *hInstance* As Integer, ByVal *Position* As Integer, ByVal *Buffer* As String) As Integer	Copies an ASCII string to an unprotected field specified by position; operation affected by SetSessionParameters() Possible return codes: 0, 1, 2, 5, 6, 7, 8, 9, 24
**ATMSetCursorLocation (ByVal *hWnd* As Integer, ByVal *Row* As Integer, ByVal *Column* As Integer) As Integer Places the host cursor at the specified row, column Possible return codes: beginning presentation space position of the string, -2, -7, -9	WD_SetCursor (ByVal *hInstance* As Integer, ByVal *Position* As Integer) As Integer	Moves the cursor to the specified position in the presentation space Possible return codes: 0, 1, 4, 7, 9, 10
ATMStartSession (ByVal *hWnd* As Integer, ByVal *Session* As String, ByVal *Visible* As String) As Integer Starts a terminal session with specified appearance; Visible = 'N' is normal, 'I' is icon, 'M' is maximized, 'H' is hidden Possible return codes: 1, -2, -9, -11, -101, -105, -200	Use EHL function: RunProfile or RDL function: RDL_RunConnect RunProfile(ByVal *Session* As String, ByVal *Visible* As String) You do not need everything else listed if you do the RDL_RunConnect.	RunProfile: Starts RUMBA with specified profile Possible return codes: 0, 2, 9
ATMStopSession (ByVal *hWnd* As Integer) As Integer Terminates the terminal session Possible return codes: 1, -2, -9, -11, -101, -104, -105	WD_DeletePS(ByVal hInstance As Integer, ByVal ShortName As String) As Integer	DeletePS: Disconnects and closes the RUMBA session. Possible return codes: 0, 1
ATMWaitForString (ByVal *hWnd* As Integer, ByVal *Row* As Integer, ByVal *Column* As Integer, ByVal *SearchString* As String, ByVal *TimeOut* As Integer) As Integer Waits until a specified string appears in the host session at row, column or until timeout occurs; Timeout is a number of half-seconds Possible return codes: beginning	Available methods: >RDL_ChangeHostScreen >SearchPS/Interruptible Pause loop >If developing in VB you would need to use the RUMBAControl to enable the AdviseHostUpdate functionality since this is a callback. >If developing any other language, can	

presentation space position of the string, 0, -1, -2, -4, -5, -7, -9, -101	use AdviseHostUpdate directly.	
ATMWaitHostQuiet (ByVal *nWnd* As Integer, ByVal *SettleTime* As Integer, ByVal *TimeOut* As Integer) As Integer Waits until the XCLOCK has disappeared from the OIA (Operator Information Area) for a specified amount of time; SettleTime is the number of seconds XCLOCK must be off before returning and TimeOut is the maximum number of half-seconds to wait Possible return codes: 1, -1, -2, -4, -5, -7, -9, -101	You can use the above noted methods or use an Interruptible Pause with SettleTime and CopyOIA.	

Attachmate Enterprise Access Library (EAL) Return Codes	**Wall-Data RUMBA Development Library (RDL) Return Codes and EHLLAPI Return Codes (*Italicized* items added to listing)**
1 = Operation successful (ATM_SUCCESS) or condition is true	>>Review Appendix C of RUMBA Tools for Visual Basic for listing of EHLLAPI Return Codes
0 = Operation unsuccessful or condition is false	0 = Function successfully completed (APIOK)
-1 = Not connected (ATM_NOTCONNECTED)	1 = Invalid session or session not active (APINOTCONNECTED)
-2 = Invalid parameter (ATM_INVALIDPARAMETER)	2 = Parameter error (APIPARAMETERERROR)
-4 = Session occupied or Time-out occurred	4 = Function successfully completed and presentation space waiting for host response (APIPSBUSY)
-5 = Session locked (ATM_SESSIONLOCKED)	5 = Input inhibited i.e. keyboard locked (APIINHIBITED)
-6 = Data was truncated or Field size mismatch	6 = Copy completed but data truncated (APITRUNCATED)
-7 = Invalid position	7 = Invalid presentation space position (APIPOSITIONERROR)
-9 = System error occurred (ATM_SYSTEMERROR)	9 = System error (APISYSERROR)
-10 = Function not available	*10 = Function not available*
-11 = Resource unavailable	11 = Resource unavailable or session already in use (APIUNAVAILABLE)
-12 = ATM session busy	*12 = The session was stopped*
-24 = Unformatted host presentation space or Search string could not be found	24 = No such string found (APINOFIELD)
-26 = Host session update	26 = Presentation space or OIA has been updated (APIPSCHANGED)
	28 = Field length had zero bytes (APIZEROLENFIELD)
-101 = Memory unavailable	
-102 = Delay ended by client	*3 = File transfer complete (APIFTXCOMPLETE)*
-103 = Unconfigured Presentation Space ID (PSID)	*4 = File transfer completed segmented (APIFTXSEGMENTED)*
-104 = No host access software attached	8 = Unavailable Operation (APINOTAVAILABLE)
-105 = Workstation Control failure	*20 = Undefined key combination (APIUNDEFINEDKEY)*
-200 = No matching Presentation Space ID (ATM_NOMATCHINGPSID)	*21 = OIA updated (APIOIAUPDATE)*
-202 = Configuration file session is open	*22 = PS updated (APIPSUPDATE)*
-301 = Event ID already in use by applications for this table ID	*23 = Both PS and OIA updated (APIBOTHUPDATE)*

-302 = Maximum number of events allowed for a table already added to this table	*25 = No keystrokes are available(APINOKEYSTROKES)*
-303 = Maximum number of tables for hWnd already added to this table	*27 = Verbose mode FTX canceled by user (APIFTXCANCELLED)*
-304 = No wait entries were added to the table	*30 = Invalid cursor type (APIINVALIDTYPE)*
-305 = The supplied event index has not been set	*31 = Keystroke overflow (APIKEYOVERFLOW)*
-306 = The row value is greater than the host session can support or not an integer greater than zero when column is zero	*302 = FTX file not found (APIFILENOTFOUND)*
-307 = The column value is greater than the host session can support or not an integer greater than zero when row is zero	*305 = FTX access denied (APIACCESSDENIED)*
-308 = WaitString longer than the maximum allowable value	*308 = FTX Insufficient memory (APIMEMORY)*
	9998 = Invalid RUMBA session ID was specified (APIINVALIDID)
	9999 = Character is data string invalid (APIINVALIDRC)