WORLDPAC
Wholesale Distributor of Original Equipment Automotive Parts

Client Automation Protocol DLL Reference

version 5.0 OCT-2022

Download

Get wcap sdk here.

Changes since version 4

We've improved wcap.dll with the addition of several new API calls to enhance the interaction between your application and speedDIAL and make debugging easier.

Autostart
WCAPConnectAS is an improved version of WCAPConnect which will, if necessary, seamlessly launch speedDIAL during an attempt to connect. This replaces three separate calls as well as a manual task switch and second button press that, in the past, were typically required of the user to accomplish the same result. The autostart feature is not available in DIAL 4.x.

Debugging
WCAPDebugEnable, WCAPDebugEnabled, and WCAPDebugWrite will help you understand the the way your application and wcap.dll work together. By enabling debug, a log file is created in your application directory which shows the activity of wcap.dll. Use WCAPDebugWrite to insert application specific information which helps you trace the flow of your program.

SDK updated
The WCAP SDK has been improved with the addition of a reference implementation of the wcap.dll API specification. See for yourself how the API is accessed.

Backward compatible
Wcap.dll is completely backward compatible with the older version of the API specification so your application can continue to work with speedDIAL while you develop new code to take advantage of the autostart features.

---

procedure WCAPRegisterReceive(RcvProcedure :FARPROC); stdcall;

Register your callback procedure with this procedure. Your called-back procedure must have the following prototype:

procedure(XMLDoc :LPSTR; Size :INT)

Use this procedure *first* before issuing any other call so that your application may get all responses DIAL sends.

More information about the role of the callback procedure in your application follows the general WCAPxxxx descriptions.

Parameters:

Var	**Type**	**Comments**
RcvProcedure	FARPROC	Address of your callback procedure.

Returns:

NONE

---

function WCAPConnectAS(Host :LPSTR; Port :UINT; Autostart :UINT) :BOOL; stdcall;

Replaces procedures WCAPStartServer() and WCAPConnect().

Use WCAPConnectAS() to initiate communication between your application and speedDIAL. The Autostart parameter (unsigned integer) will accept 0, 1, or 2.

value	meaning
0	Autostart is off. If an attempt is made to connect to speedDIAL before speedDIAL is ready to accept a connection, the attempt will fail and an error will be returned to your application
1	Autostart is on. When an attempt is made to connect to speedDIAL, speedDIAL will be launched if neccessary. Once login has happened, WCAP will attempt a connection.
2	Autostart is prompted. If speedDIAL is not yet running, the customer will be presented with a dialog asking if they wish to launch speedDIAL. The options are yes or no. If the user selects no, an error is returned to your application.

DIAL will allow a connection only from an application which is running on the SAME MACHINE as DIAL. Only 1 instance of DIAL may run on the same machine.

When WCAPConnectAS() is called, and a successful connection is established, a "Hello" document is sent to your application. This is how your application knows DIAL is ready for other WCAP activity. Simply because the result of this call is successful, it is not an indication to begin sending other WCAP requests. Please see the WCAP specification for more info on what DIAL is sending in the "Hello" reply.

Parameters:

Var	Type	Comments
Host	LPSTR	Host name
Port	UINT	Port number
Autostart	UINT	0 = no :same as WCAPConnect(). 1 = yes :launch speedDIAL, wait for login, send connect request to your application. 2 = prompt :user will be prompted by a dialog from wcap.dll to start speedDIAL.

Returns:

Type	Comments
BOOL	Returns TRUE is successful. Subsequent connect attempts to a port already in use yields FALSE

---

function WCAPConnect(Host :LPSTR; Port :UINT) :BOOL; stdcall;

(not recomended)

Use WCAPConnect() to initiate communication between your application and DIAL. To use WCAP, the user must be logged on to DIAL.

DIAL is configured to allow a connection on port 17943. The user can change this by choosing "Options | More Options | WCAP Interface" in DIAL. "Allow connection" must be checked, or DIAL will not allow a connection.

DIAL will allow a connection only from an application which is running on the SAME MACHINE as DIAL. Only 1 instance of DIAL may run on the same machine.

If the connection attempt is successful, a Hello document is sent to your application. Otherwise, you will receive a Error document.

Parameters:

Var	**Type**	**Comments**
Host	LPSTR	Host name
Port	UINT	Port number

Returns:

Type	**Comments**
BOOL	Returns TRUE is successful. Subsequent connect attempts to a port already in use yields FALSE

---

function WCAPStartServer :BOOL; stdcall;

(not recomended)

Use this function to start DIAL or speedDIAL. This function is not recommended. Use WCAPConnectAS instead to start speedDIAL automatically.

Parameters:

NONE

Returns:

Type	**Comments**
BOOL	Returns TRUE is successful launch of DIAL or speedDIAL

---

function WCAPConnected :BOOL; stdcall;

Test whether a connection with DIAL is established. You can use this function before transmitting WCAP request documents. UPDATED: This function has been improved to provide better reliability.

Parameters:

NONE

Returns:

Type	Comments
BOOL	Returns TRUE if an active connection is detected

---

procedure WCAPDisconnect; stdcall;

End an active connection with DIAL. It's important that WCAPDisconnect() is issued BEFORE your application is destroyed. Otherwise, unfriendly Windows messages appear to your user.

Parameters:

NONE

Returns:

NONE

---

procedure WCAPSend(XMLDoc :LPSTR); stdcall;

You construct a string representation of the WCAP request and pass it to this procedure. You are responsible for allocating and freeing the memory associated with it.

Parameters:

Var	Type	Comments
XMLDoc	LPSTR	Long pointer to a null-terminated string of a complete document request

Returns:

NONE

procedure WCAPDebugEnable(EnableLevel :UINT); stdcall;

Start debugging WCAP by setting EnableLevel to 0, 1, or 2.
Level 1 will record the XML traffic in/out bound. Level 2 will record everything level 1 does in addition to API calls and the various internal events which make wcap work.

Setting the EnableLevel to 0 stops recording.

Debug information is found in your application folder in file wcapdebug.txt.

Parameters:

Var	**Type**	**Comments**
EnableLevel	UINT	Debug level: 0 - off 1 - debug on 2 - debug extreme

function WCAPDebugEnabled :UNIT; stdcall;

Returns current debug level.

Parameters:

NONE

Returns:

Type	**Comments**
UINT	Current debug level

procedure WCAPDebugWrite(DebugString :LPSTR); stdcall;

User supplied text string to assist in trace.

Parameters:

Var	Type	Comments
DebugString	LPSTR	Pointer to a string containing the text you want written to wcapdebug.txt

---

Callback Procedure

Your application will require a callback procedure prototyped as:

procedure(XMLDoc :LPSTR; Size :UINT)

Parameters:

Var	Type	Comments
XMLDoc	LPSTR	Long pointer to a null-terminated string of a complete document response
Size	INT	Integer size of the string (not including the null)

Pass the address of your callback procedure to *WCAPRegisterReceive()* function so your application can receive document events from DIAL.

WCAP.DLL will use this procedure to send **complete** WCAP XML documents to your application. Each time your application receives a document with this callback you may assume:

- One document event has occurred
- Only one document is pointed to in the incoming parameter
- Memory for the parameter is allocated in the dll and is freed upon return from the call
- If you need the save the document, it's up to you to create a new variable and copy the data