IP Office CTI Link

DevLink Programmer's Guide

40DHB0002UKAD Issue 11a (14th June 2005)

Table Of Contents

CTI DevLink

Overview

IP Office Dev*Link* is part of the IP Office CTI Link Software Development Kit. The IP Office CTI Link is available in Lite and Pro versions, which provide run-time interfaces for applications to use. The Software Development Kit (SDK) provides documentation on both Lite and Pro interfaces for software developers.

Both the Lite and Pro offerings are the same program. The additional functionality provided by IP Office CTI Link Pro is enabled when the CTI Link Pro licence key is installed.

This manual provides a tutorial and reference to the Dev*Link* Pro interface, as well as providing example source code.

IP Office CTI Link Lite

IP Office CTI Link Lite is a free of charge offering, and contains the following component:

- **TAPI*Link* Lite**
  This component implements Microsoft TAPI, and allows programs to control one telephone line per PC. It provides simple CTI, including answer, hangup, make-call, transfer, and provides CLI/ANI for screen-popping. Please see the TAPI*Link* Developers Guide for more details on TAPI*Link* Lite and TAPI*Link* Pro.

Note that the first version of CTI Link Lite contained DevLink Lite (DevLink version 1.0.0.3) which provided a Call Logging interface. This has been superseded by IP Office SMDR, which is available on the IP Office 1.3 Admin CD. DevLink version 1.0.0.4 no longer exports the Call Logging interface.

IP Office CTI Link Pro

IP Office CTI Link Pro includes all of the Lite functionality and is accessed via a licence key. It contains the following components:

- **TAPI*Link* Pro**
  This component provides both first-party and third-party TAPI control of telephony devices. In addition to the functionality provided by TAPILink Lite, it also adds the ability to receive information on ACD queues, hunt groups, and provides additional advanced functionality. Please see the TAPI*Link* Developers Guide for more details on TAPI*Link* Lite and TAPI*Link* Pro.
- **Dev*Link* Pro**
  This component provides a real-time event stream. The real-time event stream provides information on telephone activity as and when that activity occurs, and also provides information on trunk activity.

DevLink

Dev*Link* provides a programming interface which complements the Microsoft TAPI interfaces provided by TAPI*Link* Lite and Pro:

- Real-time event stream
  The Real-time event stream is enabled by installing the CTI Pro licence key onto the system unit.

The Dev*Link* DLL, which is included on the User CD-ROM should be installed with the licence key. Dev*Link* enables third-party applications, such as call recorders to use information provided by the telephone system. Please refer to the IP Office CTI Link Installation Manual for installation instructions.

When the Dev*Link* component is installed, a Windows Dynamic Link Library, DEVLINK.DLL is installed, by default, into "Program Files/Avaya/IP Office/DEV Link" directory. Programs using this interface simply need to link to this library and use it's functions.

Notes:

1. Any application that uses the Dev*Link* DLL should include it in the application installation script. As the DLL is not a COM component, no registration is necessary. The DLL does not require any registry settings or supporting files.
2. When developing an application that uses the DLL, the Dev*Link* header file (devlink.h) and appropriate library file (devlink.lib or devlinkomf.lib) should be copied to the same directory as the project source files. The devlink.lib library file contains export symbols in COFF format which can be used with projects written in Visual C++. The devlinkomf.lib library file contains export symbols in OMF format for other linkers.

Using the DevLink DLL

DEVLINK.DLL can be used in any language that supports Dynamic Link Libraries (DLLs), e.g. Microsoft Visual C++ or Borland Delphi.

Dev*Link* allows developers to request call-backs, which will be generated when significant events occur. For the real-time event stream, this occurs at various stages throughout a call's lifetime. Both telephony calls and data calls are included in the events generated.

Please note that all application call-backs made on a thread which Dev*Link* creates. As a result, consideration must be given by programmers to ensure that all functions called from within a call-back are thread-safe.

Using DevLink with Microsoft Visual C++

Appendix A contains the DEVLINK.H file which should be used with Microsoft Visual C++. Programs written in Microsoft Visual C++ should link to the DEVLINK.LIB library.

Using DevLink with Borland Delphi

Appendix B contains the DEVLINK.PAS file which should be used with Borland Delphi. Programs written using Borland Delphi should use DEVLINK.PAS, which links directly to the DEVLINK.DLL library.

Connecting to an IP Office using DevLink

Connecting

Dev*Link* supports connection to multiple IP Office systems at the same time. To connect to an IP Office system, the `DLOpen()` function must be called:

```
LONG DLOpen(
LONG pbxh,
TEXT *pbx_address,
TEXT *pbx_password,
TEXT *reserved1,
TEXT *reserved2,
COMMSEVENT cb);
```

The application-supplied pbxh handle is used internally by Dev*Link* to track the connected IP Office System. Each connected system must have a different pbxh handle, supplied by the application.

The pbx_address field is the IP address of the IP Office system unit. A value of "255.255.255.255" can be used, in which case DevLink will broadcast to locate an IP Office system unit.

Notes:

1. If Dev*Link* is being used to control more than one IP Office system at the same time, then the specific IP address of the IP Office **must** be used.
2. The cb parameter (Communications Status Callback) is required, and must not be set to NULL. The return result from DLOpen () does not necessarily indicate whether or not the connection to the system unit was successful. If the connection attempt succeeds, then a COMMSEVENT callback will be generated, indicating that connection to the system has succeeded.
3. The pbx_password parameter should be the monitor password of the switch, not the system password.
4. The reserved1 and reserved2 parameters are for future expansion, and should be set to NULL (nil in Delphi).

Example: Connecting to IP Office in "C"

Note that the "systempassword" in the call to DLOpen () should be replaced with your unit's actual system password.

```
#include <windows.h>
#include <stdio.h>
#include "devlink.h"
LONG hEvent;
DWORD dwCommsEvent;
BOOL bStarting;
void CALLBACK HandleCommsEvent( LONG pbxh, DWORD comms_evt, DWORD parm1 )
{
switch( comms_evt ) {
case DEVLINK_COMMS_OPERATIONAL:
// we are working fine... fall through
case DEVLINK_COMMS_NORESPONSE:
// system not found (initial connection),
// or network connection lost (rebooted?)
// fall through...
case DEVLINK_COMMS_REJECTED:
// incorrect system password specified...
if( bStarting ) {
dwCommsEvent = comms_evt;
SetEvent( hEvent );
}
else {
// insert your code here...
}
break;
case DEVLINK_COMMS_MISSEDPACKETS:
// Indicates that the system is under
// heavy load. IP Office always prioritises
// data routing and call handling above CTI events.
// (parm1 contains the number of packets missed)
break;
}
}
int main(int argc, char* argv[])
{
printf( "connecting...");
bStarting = TRUE;
hEvent = CreateEvent( NULL, FALSE, FALSE, NULL );
DLOpen( 0,
"255.255.255.255"
"systempassword",
NULL,
NULL,
HandleCommsEvent );
dwCommsEvent = DEVLINK_COMMS_NORESPONSE;
WaitForSingleObject( hEvent, 10000 ); // 10 seconds
bStarting = FALSE;
if( dwCommsEvent == DEVLINK_COMMS_OPERATIONAL ) {
printf("Connected OK\n");
}
else {
printf("Error connecting to IP Office\n");
}
DLClose( 0 );
CloseHandle( hEvent );
return 0;
}
```

Example: Connecting to IP Office in Delphi

Note that the "systempassword" in the call to DLOpen () should be replaced with your unit's actual system password.

```
unit Unit1;
interface
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls;
type
TForm1 = class(TForm)
Button1: TButton;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1;
implementation
uses
devlink;
{$R *.DFM}
var
hEvent : THANDLE;
dwCommsEvent : DWORD;
bStarting: boolean;
procedure HandleCommsEvent( pbxh : LongInt;
Comms_status : DWORD;
Parm1 : DWORD ); stdcall;
begin
case Comms_status of
DEVLINK_COMMS_OPERATIONAL,
DEVLINK_COMMS_NORESPONSE, DEVLINK_COMMS_REJECTED:
begin
if bStarting then
begin
dwCommsEvent := comms_status;
SetEvent( hEvent );
end;
end;
DEVLINK_COMMS_MISSEDPACKETS:
begin
// parm1 indicates the number of packets missed...
end;
end;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
bStarting := TRUE;
hEvent := CreateEvent( nil, FALSE, FALSE, nil );
DLOpen( 0, '255.255.255.255', 'systempassword', nil, nil, - HandleCommsEvent
);
dwCommsEvent := DEVLINK_COMMS_NORESPONSE;
WaitForSingleObject( hEvent, 10000 ); // 10-second timeout
bStarting := FALSE;
if dwCommsEvent = DEVLINK_COMMS_OPERATIONAL then
begin
ShowMessage('Connected OK');
end
else
begin
```

```
ShowMessage('Error connecting to IP Office');
end;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
DLClose( 0 );
CloseHandle( hEvent );
end;
end.
```

Disconnecting

To disconnect from IP Office, use the DLClose() routine, passing the same application-supplied handle that was used to open the connection:

```
LONG PASCAL DLClose( LONG pbxh );
```

DevLink Real-Time Event Stream

DevLink Real-Time Event Stream

Calls in IP Office are modelled as being a communications line between two end-points, called A and B respectively. An A end is always present, but a B end may or may not be present, depending on the state of the call (A and B are typically extensions or trunks, but they may also be connected to the voice mail system or parked).

Three different types of real-time events are generated by Dev*Link*. These are used to track the call throughout its lifetime:

- **S events**
  S events give information on the status of a call. S events are generated when calls are first created, answered, or the status of a device involved in the call changes.
- **D events**
  D events are generated when the call is completed or abandoned. They indicate that the call no longer exists.
- **A events**
  A events are generated when one end of a call is connected to a line (such as an ISDN, QSig or VoIP line) and the IP Office decides to swap the A end and the B end of the call. Examples of when this may happen include;
  - When a parked party hangs up,
  - When an outgoing call is transferred,
  - When a call is un-parked.

The format and information contained in these events are described in more detail below.

Applications request information real-time events by calling the DLRegisterCallDelta2() function after they have connected to the IP Office system:

```
LONG PASCAL DLRegisterType2CallDeltas( HANDLE pbxh, CALLLOGEVENT cb );
```

This registers a function provided by the application, of type CALLLOGEVENT, which will be called by Dev*Link* whenever a real-time event is generated.

```
typedef void (CALLBACK * CALLLOGEVENT)(
LONG pbxh,
TEXT * info
);
```

The two parameters (pbxh and info) are provided to the application. The pbxh parameter is the identifier that was used in the call to DLOpen() and can be used to identify the system when the application is connected to multiple IP Office systems.

The second parameter is a string, containing the call record. The call record is a comma-separated string, with variable width fields. The string will always be less than 1500 bytes in length inclusive of a terminating NULL character.

S events

S events are sent whenever a call is first created, and whenever any part of the call changes state.

The format of an S event is as follows:

```
    CALL: S <field1>,<field2>, …<field50>
```

All of the information is provided in the character string. Thus, numbers are represented in ASCII form. All numbers are integers, and do not contain a decimal point. Each string field has a maximum of 128 characters in length.

The field definitions are documented in the following table. Please note that fields marked as <reserved> may contain information, but their contents and meaning will change between releases of the IP Office core software.

Field	Name	Type	Description
1	A call id	String	Call id for the A end of the call
2	B calld id	String	Call id for the B end of the call
3	Astate	Number	State of the A end of the call. Valid numbers are: • 0 Idle • 1 Ringing • 2 Connected • 3 Disconnected • 4 Suspending • 5 Suspended • 6 Resuming • 7 Dialling • 8 Dialled • 9 Local Dial • 10 Queued • 11 Parked • 12 Held • 13 Redialling
4	Bstate	Number	State of the B end of the call – see above for values. If there is no B end, this will be 0 (Idle)
5	Aconnected	Number	0 indicates not connected to the switch (e.g. If receiving dialtone)
6	A is music	Number	Indicates whether the A end of the call is listening to music on hold
7	Bconnected	Number	As in field 5 above, but for the B end
8	B is music	Number	Indicates whether the B end of the call is listening to music on hold
9	Aname	String	Name of the A end of the call. The format of this string is: For extensions: Name (Number) For trunks: Line number
10	Bname	String	As in field 9 above. If there is no B end, this will be empty.
11	Blist	String	List of possible targets for the call. Targets are separated by semicolons, and if there are more than four, the list is terminated after 4 elements with an ellipsis.
12	Aslot Achannel	String	Aslot is the slot number of the A side of the call and is the identify of the TDM trunk. Achannel is the channel number within the Aslot. The format of the string is "Number.Number".
13	Bslot Bchannel	String	Bslot is the slot number of the B side of the call and is the identify of the TDM trunk. Bchannel is the channel number within the Bslot. The format of the string is "Number.Number".
14	Called party presentation &	String	This field contains two numbers: 1) an indicator of how the called party details should be presented, and 2) the type of data to be

	type		found in the called party number field. The format of the string is "Number.Number".
15	Called party number	String	The identifier or number of the called party.
16	Calling party presentation & type	String	This field contains two numbers: 1) an indicator of how the calling party details should be presented, and 2) the type of data to be found in the calling party number field. The format of the string is "Number.Number".
17	Calling party number	String	The identifier or number of the calling party.
18	Called sub address	String	The sub address of the called party.
19	Calling sub address	String	The sub address of the calling party.
20	Dialled party type	Number	The type of data to be found in the dialled party number field.
21	Dialled party number	String	The identifier or number of the dialled party.
22	Keypad type	Number	The type of data to be found in the keypad number field.
23	Keypad number	String	The dialled digits.
24	Ring attempt count	Number	The number of times this call has been presented to a target.
25	Cause	Number	The reason the call is in it's current state. See the table below for further details.
26	Voicemail disallow	Number	1 if the call cannot divert to voice mail.
27	Sending complete	Number	1 if overlap dialling is complete.
28	Call type & - Transport type	String	This field contains two numbers: 1) call type, e.g. speech, video and 2) the transport type, 0=circuit, 2=packet. The format of the string is "Number.Number".
29	Owner hunt group name	String	The name of the hunt group where the call is currently queued.
30	Original hunt group name	String	The name of the hunt group where the call was originally targeted.
31	Original user name	String	The name of the user where the call was originally targeted.
32	Target hunt group name	String	The name of the hunt group where the call has been targeted at.
33	Target user name	String	Name of user to which the call is being targeted.
34	Target RAS name	String	The name of the internal port where the call is being targeted.
35	IsInternalCall	Number	1 the call is internal, otherwise 0.
36	Time stamp	String	The time the call was created (internal tick count).
37	Connected time	Number	The time the call was connected.
38	Ring time	Number	The time the call started ringing.
39	Connected duration	Number	The duration the call has been connected or was connected for.
40	Ring duration	Number	The duration the call has been ringing or was ringing for.
41	Locale	String	Locale identifier (same as the locale setting in Manager).
42	ParkslotNumber	String	When the call is parked, the B end is undefined, and this field contains the park slot number.
43	Call waiting	String	1 if the call is a call waiting call
44	Tag	String	Contains the tag, which may be applied through TAPI, Voice Mail/Pro or SoftConsole.
45	Transferring	Number	Non-zero indicates that the call is being transferred.
46	Service active	Number	Non-zero if the service is active.
47	Service quota	Number	Non-zero if the service quota is used.

	used		
48	Service quota time	Number	The service quota time.
49	Account code	String	The account code of the call, if any
50	CallID	Number	Unique call id

Field 25 Cause Codes

0	CMCauseUnknown	
1	CMCauseUnallocatedNumber	The number is not in the dial plan
2	CMCauseForceIdle	Force clear down unconditionally
3	CMCauseUnregister	Indicate the operation is for unregistration procedure
16	CMCauseNormal	
17	CMCauseBusy	The call target is busy
18	CMCauseNoUserResponding	
21	CMCauseCallRejected	The call has not been permitted, e.g. call barring
31	CMCauseNormalUnspecified	
34	CMCauseNoChannel	There is no available external line
38	CMCauseNetworkOOO	Problem with the external network
88	CMCauseIncompatible	
113	CMCausePhoneInfo	
114	CMCauseReminderFree	The call has returned because it was parked against a busy extension that is now free
115	CMCauseReminderNoAns	The call has returned because it was not answered within the transfer return timeout
116	CMCauseE911Emergency	
117	CMCauseParked	
118	CMCauseUnParked	
119	CMCausePickup	
120	CMCauseReminder	A held or parked call is returning due to a hold / park timeout
121	CMCauseRedirect	The call has been redirected
122	CMCauseCallBarred	The call was barred
123	CMCauseForwardToVoicemail	The call has been forwarded to voicemail
124	CMCauseAnsweredByOther	The call has been answered by someone else
125	CMCauseNoAccountCode	The call has not been allowed because a valid account code has not been provided
126	CMCauseTransfer	The call has been transferred
127	CMCauseConferencingMove	The call has been placed into a conference
128	CMCauseRestrictedToPartner	
129	CMCauseHeldCall	
130	CMRingBackCheck	
131	CMCauseAppearanceCallSteal	The call was answered on an appearance button on another phone
132	CMCauseAppearanceBridgeInto	
133	CMCauseBumpedCall	The call is no longer the primary call, i.e. it is now a waiting call
134	CMCauseLineAppearanceCall	
135	CMCauseUnheldCall	
136	CMCauseReplaceCurrentCall	
137	CMCauseGlare	
138	CMCauseR21CompatConfMove	The call has been placed into a conference

D events

D events signify that the call is deleted. The format of a D event is as follows:

```
CALL: D <field1>,<field2>,…<field3>
```

The fields are:

Field	Name	Type	Description
1	A call id	String	Call id for the A end of the call
2	B call id	String	Call id for the B end of the call
3	CallID	Number	Unique call id

A events

A events indicate that the call ends have been swapped. This occurs, for example, when the originating extension unparks an external call. The format of an A event is very similar to that for a D event:

```
CALL: A <field1>,<field2>,…<field3>
```

The fields are:

Field	Name	Type	Description
1	A call id	String	Call id for the A end of the call
2	B call id	String	Call id for the B end of the call
3	CallID	Number	Unique call id

DevLink reference

Functions

DLOpen

The DLOpen() routine is used to connect to an IP Office system.

Parameters

- **pbxh** - A number used to identify the system. This is a user-supplied parameter that must remain consistent across all calls to Dev*Link*.
- **pbx_address** - The IP address of the IP Office system (either a IP address or a host name can be used). This may be set to "255.255.255.255" in which case Dev*Link* will make a network broadcast to locate the system unit. Please note that only the first unit to respond will be connected to; if you wish to connect to multiple system units, you must specify the IP address or host name of the unit here.
- **pbx_password -** The password of the IP Office system.
- **reserved1** - This parameter should be set to NULL (nil in Delphi)
- **reserved2 -** This parameter should be set to NULL (nil in Delphi)
- **cb** - This is the address of a call-back function, to which connection status events will be sent. This parameter is mandatory.

Return value

This routine may return either 0 (DEVLINK_SUCCESS) or 1 (DEVLINK_UNSPECIFIEDFAIL).

Note that a return value of DEVLINK_SUCCESS only indicates that communications with the unit has been initiated; the subsequent connection may fail for several reasons. Further information will be provided to the COMMSEVENT callback function specified in the cb parameter.

C / C++

```
LONG PASCAL DLOpen( LONG pbxh
, TEXT * pbx_address
, TEXT * pbx_password
, TEXT * reserved1
, TEXT * reserved2
, COMMSEVENT cb
);
```

Delphi

```
function DLOpen(pbxh: LongInt;
pbx_address: PChar;
pbx_password: PChar;
reserved1: PChar;
reserved2: PChar;
cb: TCommsEvent): LongInt; stdcall;
```

DLClose

The DLClose() routine is used to disconnect from an IP Office system.

Parameters

- **pbxh -** A number used to identify the system. This is the user-supplied parameter used to connect to Dev*Link* in the call to DLOpen().

Return value

- This routine may return 0 (DEVLINK_SUCCESS) or 1 (DEVLINK_UNSPECIFIEDFAIL) in the event of an error.

C / C++

```
LONG PASCAL DLClose( LONG pbxh );
```

Delphi

```
function DLClose(pbxh: LongInt): LongInt; stdcall;
```

DLRegisterType2CallDeltas

The DLRegisterType2CallDeltas() routine is used to request Call Delta information.

Parameters

- **pbxh -** A number used to identify the system. This is the user-supplied parameter used to connect to Dev*Link* in the call to DLOpen().
- **cb -** The address of the callback function that will receive real-time events. Only one callback can receive real-time events at one time, and if this parameter is NULL, then real-time events will no longer be sent to the application.

Return value

- This routine may return:-
  - 0 = DEVLINK_SUCCESS
  - 1 = DEVLINK_UNSPECIFIEDFAIL - Returned in the event of an error.
  - 2 = DEVLINK_LICENCENOTFOUND - If no CTI licence is activated on the IP Office system.

C / C++

```
LONG PASCAL DLRegisterType2CallDeltas( LONG pbxh, CALLLOGEVENT cb );
```

Delphi

```
function DLRegisterType2CallDeltas(pbxh: LongInt;
cb: TCallLogEvent): LongInt; stdcall;
```

Callbacks

COMMSEVENT

The COMMSEVENT callback is called by Dev*Link* whenever the state of the communication with the IP Office system unit changes.

Parameters

- **pbxh -** A number used to identify the system. This is the user-supplied parameter used to connect to the IP Office system unit in DLOpen().
- **comms_state** - A number indicating the state of the communications. Valid values are:

Name	Value	Description
DEVLINK_COMMS_OPERATIONAL	0	Communications established. This occurs either after the initial call to DLOpen(), or after the system unit has come back on-line after being powered off or rebooted.
DEVLINK_COMMS_NORESPONSE	1	No response from system unit. This occurs either after the initial call to DLOpen(), or if the system unit is powered off or rebooted. It can also occur if network problems prevent communications.
DEVLINK_COMMS_REJECTED	2	Reserved for future use
DEVLINK_COMMS_MISSEDPACKETS	3	Packets were generated by the IP Office system unit, but were not received by Dev*Link*. This can occur either because the IP Office system unit is under heavy load, or because the application using Dev*Link* did not return from a callback quickly enough. Applications should ensure that they do not take more than 100 milliseconds to process events.

- **parm1 -** This value is only defined for: DEVLINK_COMMS_MISSEDPACKETS events, in which case it indicates the number of packets dropped.

Return value

- No return value.

C / C++

```
typedef void (CALLBACK * COMMSEVENT)(
LONG pbxh,
DWORD comms_state,
DWORD parm1
);
```

Delphi

```
type
TCommsEvent = procedure( pbxh : LongInt;
comms_state : DWORD;
Parm1 : DWORD );
```

CALLLOGEVENT

The CALLLOGEVENT callback is called by Dev*Link* to deliver a real-time (Delta2) event.

- **Note:** A CTI license is required for returning Delta2 events.

Parameters

pbxh - A number used to identify the system. This is the user-supplied parameter used to connect to the IP Office system unit in DLOpen().

info - Text string containing the event. Please see the previous section on real-time events for more details.

Return value

- No return value.

C / C++

```
typedef void (CALLBACK * CALLLOGEVENT)(
Long pbxh,
TEXT * info
);
```

Delphi

```
type
TCallLogEvent = procedure( pbxh : LongInt; info : PChar );
```

Appendices

DEVLINK.PAS

This appendix contains a copy of the DEVLINK.PAS file, used for Borland Delphi programs.

```
unit DEVLINK;
{**************************************************************************}
{ Delphi unit for DevLink (c) 2001 Avaya Global SME Solutions   }
{ Contents:- }
{ IP Office DevLink DLL provides an interface for managing }
{ the IP Office product ranges from a Windows PC }
{**************************************************************************}
interface
uses
Windows;
const
DEVLINK_SUCCESS = 0;
DEVLINK_UNSPECIFIEDFAIL = 1;
DEVLINK_LICENCENOTFOUND = 2;
const
DEVLINK_COMMS_OPERATIONAL = 0;
DEVLINK_COMMS_NORESPONSE = 1;
DEVLINK_COMMS_REJECTED = 2;
DEVLINK_COMMS_MISSEDPACKETS = 3;
type
TCallLogEvent = procedure( pbxh : LongInt; info : PChar ); stdcall;
type
TCommsEvent = procedure( pbxh : LongInt;
Comms_status : DWORD;
Parm1 : DWORD ); stdcall;
function DLOpen(pbxh: LongInt;
pbx_address: PChar;
pbx_password: PChar;
reserved1: PChar;
reserved2: PChar;
cb: TCommsEvent): LongInt; stdcall;
function DLClose(pbxh: THandle): LongInt; stdcall;
function DLRegisterType2CallDeltas(pbxh: LongInt;
cb: TCallLogEvent): LongInt; stdcall;
implementation
function DLOpen; external 'DEVLINK.DLL';
function DLClose; external 'DEVLINK.DLL';
function DLRegisterType2CallDeltas; external 'DEVLINK.DLL';
end.
```

DEVLINK.H

This appendix contains a copy of the DEVLINK.H header file, used for C and C++ programs.

```
/*******************************************************************/
/* */
/* C/C++ Header File (c) 2001 Avaya Global SME Solutions */
/* */
/* Contents:- */
/* IP Office Dev link DLL provides an interface for managing  */
/* the IP Office product ranges from a Windows PC.  */
/*******************************************************************/
#ifndef _DEVLINK_H_
#define _DEVLINK_H_
typedef char TEXT;
#define DEVLINK_SUCCESS - 0
#define DEVLINK_UNSPECIFIEDFAIL 1
#define DEVLINK_LICENCENOTFOUND 2
#define DEVLINK_COMMS_OPERATIONAL  0
#define DEVLINK_COMMS_NORESPONSE  1
#define DEVLINK_COMMS_REJECTED  2
#define DEVLINK_COMMS_MISSEDPACKETS  3
#ifdef __cplusplus
extern "C"
{
#endif
typedef void (CALLBACK * CALLLOGEVENT)(
LONG pbxh,
TEXT * info
);
typedef void (CALLBACK * COMMSEVENT)(
LONG pbxh,
DWORD comms_state,
DWORD parm1
);
LONG PASCAL DLOpen( HANDLE pbxh
, TEXT * pbx_address
, TEXT * pbx_password
, TEXT * reserved1
, TEXT * reserved2
, COMMSEVENT cb
);
LONG PASCAL DLClose( LONG pbxh );
LONG PASCAL DLRegisterType2CallDeltas( LONG pbxh, CALLLOGEVENT cb );
#ifdef __cplusplus
};
#endif
#endif // _DEVLINK_H_
```

Index

Avaya
Sterling Court
15 - 21 Mundells
Welwyn Garden City
Hertfordshire
AL7 1LZ
England

Tel: +44 (0) 1707 392200
Fax: +44 (0) 1707 376933

Web: http://www.avaya.com