



# **IP Office**

## **DevLink Programmer's Guide**

#### Notices

While reasonable efforts have been made to ensure that the information in this document is complete and accurate at the time of printing, Avaya assumes no liability for any errors. Avaya reserves the right to make changes and corrections to the information in this document without the obligation to notify any person or organization of such changes.

#### Documentation disclaimer

Avaya shall not be responsible for any modifications, additions, or deletions to the original published version of this documentation unless such modifications, additions, or deletions were performed by Avaya.

End User agree to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation, to the extent made by End User.

#### Link disclaimer

Avaya is not responsible for the contents or reliability of any linked Web sites referenced within this site or documentation(s) provided by Avaya. Avaya is not responsible for the accuracy of any information, statement or content provided on these sites and does not necessarily endorse the products, services, or information described or offered within them. Avaya does not guarantee that these links will work all the time and has no control over the availability of the linked pages.

#### Warranty

Avaya provides a limited warranty on this product. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya's standard warranty language, as well as information regarding support for this product, while under warranty, is available to Avaya customers and other parties through the Avaya Support Web site: <http://www.avaya.com/support>. Please note that if you acquired the product from an authorized Avaya reseller outside of the United States and Canada, the warranty is provided to you by said Avaya reseller and not by Avaya.

#### Licenses

THE SOFTWARE LICENSE TERMS AVAILABLE ON THE AVAYA WEBSITE, [HTTP://SUPPORT.AVAYA.COM/LICENSEINFO/](http://support.avaya.com/licenseinfo/) ARE APPLICABLE TO ANYONE WHO DOWNLOADS, USES AND/OR INSTALLS AVAYA SOFTWARE, PURCHASED FROM AVAYA INC., ANY AVAYA AFFILIATE, OR AN AUTHORIZED AVAYA RESELLER (AS APPLICABLE) UNDER A COMMERCIAL AGREEMENT WITH AVAYA OR AN AUTHORIZED AVAYA RESELLER. UNLESS OTHERWISE AGREED TO BY AVAYA IN WRITING, AVAYA DOES NOT EXTEND THIS LICENSE IF THE SOFTWARE WAS OBTAINED FROM ANYONE OTHER THAN AVAYA, AN AVAYA AFFILIATE OR AN AVAYA AUTHORIZED RESELLER, AND AVAYA RESERVES THE RIGHT TO TAKE LEGAL ACTION AGAINST YOU AND ANYONE ELSE USING OR SELLING THE SOFTWARE WITHOUT A LICENSE. BY INSTALLING, DOWNLOADING OR USING THE SOFTWARE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF AND THE ENTITY FOR WHOM YOU ARE INSTALLING, DOWNLOADING OR USING THE SOFTWARE (HEREINAFTER REFERRED TO INTERCHANGEABLY AS "YOU" AND "END USER"), AGREE TO THESE TERMS AND CONDITIONS AND CREATE A BINDING CONTRACT BETWEEN YOU AND AVAYA INC. OR THE APPLICABLE AVAYA AFFILIATE ("AVAYA").

Avaya grants End User a license within the scope of the license types described below. The applicable number of licenses and units of capacity for which the license is granted will be one (1), unless a different number of licenses or units of capacity is specified in the Documentation or other materials available to End User. "Designated Processor" means a single stand-alone computing device. "Server" means a Designated Processor that hosts a software application to be accessed by multiple users. "Software" means the computer programs in object code, originally licensed by Avaya and ultimately utilized by End User, whether as stand-alone products or pre-installed on Hardware. "Hardware" means the standard hardware originally sold by Avaya and ultimately utilized by End User.

#### License types

Designated System(s) License (DS). End User may install and use each copy of the Software on only one Designated Processor, unless a different number of Designated Processors is indicated in the Documentation or other materials available to End User. Avaya may require the Designated Processor(s) to be identified by type, serial number, feature key, location or other specific designation, or to be provided by End User to Avaya through electronic means established by Avaya specifically for this purpose.

#### Copyright

Except where expressly stated otherwise, no use should be made of materials on this site, the Documentation(s) and Product(s) provided by Avaya. All content on this site, the documentation(s) and the product(s) provided by Avaya including the selection, arrangement and design of the content is owned either by Avaya or its licensors and is protected by copyright and other intellectual property laws including the sui generis rights relating to the protection of databases. You may not modify, copy, reproduce, republish, upload, post, transmit or distribute in any way any content, in whole or in part, including any code and software. Unauthorized reproduction, transmission, dissemination, storage, and or use without the express written consent of Avaya can be a criminal, as well as a civil, offense under the applicable law.

#### Third Party Components

Certain software programs or portions thereof included in the Product may contain software distributed under third party agreements ("Third Party Components"), which may contain terms that expand or limit rights to use certain portions of the Product ("Third Party Terms"). Information regarding distributed Linux OS source code (for those Products that have distributed the Linux OS source code), and identifying the copyright holders of the Third Party Components and the Third Party Terms that apply to them is available on the Avaya Support Web site: <http://support.avaya.com/Copyright>.

#### Preventing toll fraud

"Toll fraud" is the unauthorized use of your telecommunications system by an unauthorized party (for example, a person who is not a corporate employee, agent, subcontractor, or is not working on your company's behalf). Be aware that there can be a risk of toll fraud associated with your system and that, if toll fraud occurs, it can result in substantial additional charges for your telecommunications services.

#### Avaya fraud intervention

If you suspect that you are being victimized by toll fraud and you need technical assistance or support, call Technical Service Center Toll Fraud Intervention Hotline at +1-800-643-2353 for the United States and Canada. For additional support telephone numbers, see the Avaya Support Web site: <http://support.avaya.com>. Suspected security vulnerabilities with Avaya products should be reported to Avaya by sending mail to: [securityalerts@avaya.com](mailto:securityalerts@avaya.com).

#### Trademarks

Avaya and Aura are trademarks of Avaya, Inc. The trademarks, logos and service marks ("Marks") displayed in this site, the documentation(s) and product(s) provided by Avaya are the registered or unregistered Marks of Avaya, its affiliates, or other third parties. Users are not permitted to use such Marks without prior written consent from Avaya or such third party which may own the Mark. Nothing contained in this site, the documentation(s) and product(s) should be construed as granting, by implication, estoppel, or otherwise, any license or right in and to the Marks without the express written permission of Avaya or the applicable third party. Avaya is a registered trademark of Avaya Inc. All non-Avaya trademarks are the property of their respective owners.

#### Downloading documents

For the most current versions of documentation, see the Avaya Support Web site: <http://www.avaya.com/support>

#### Contact Avaya Support

Avaya provides a telephone number for you to use to report problems or to ask questions about your product. The support telephone number is 1-800-242-2121 in the United States. For additional support telephone numbers, see the Avaya Web site: <http://www.avaya.com/support>

---

# Contents

## 1. CTI DevLink

|  |    |
|--|----|
| 1.1 DevLink DLL.....                           | 8  |
| 1.2 Connecting to IP Office using DevLink..... | 9  |
| 1.3 Example: Connecting in Delphi.....         | 10 |
| 1.4 Example: Connecting in C.....              | 11 |
| 1.5 Disconnecting.....                         | 12 |

## 2. DevLink Real-Time Event Stream

|                    |    |
|--------------------|----|
| 2.1 S events ..... | 15 |
| 2.2 D events ..... | 18 |
| 2.3 A events ..... | 18 |

## 3. Functions

|                                    |    |
|------------------------------------|----|
| 3.1 DLOpen .....                   | 20 |
| 3.2 DLClose .....                  | 21 |
| 3.3 DLRegisterType2CallDeltas..... | 21 |

## 4. Callbacks

|                       |    |
|-----------------------|----|
| 4.1 COMMSEVENT.....   | 24 |
| 4.2 CALLLOGEVENT..... | 25 |

## 5. Appendices

|                                   |    |
|-----------------------------------|----|
| 5.1 Appendix B: DEVLINK.PAS ..... | 28 |
| 5.2 Appendix A: DEVLINK.H.....    | 29 |
| Index .....                       | 31 |



# Chapter 1.

# CTI DevLink



---

# 1. CTI DevLink

IP Office DevLink 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 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 DevLink 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:

- **TAPILink 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 TAPILink Developers Guide for more details on TAPILink Lite and TAPILink Pro.
  - 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:

- **TAPILink 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 TAPILink Developers Guide for more details on TAPILink Lite and TAPILink Pro.
- **DevLink Pro**  
DevLink provides a programming interface which complements the Microsoft TAPI interfaces provided by TAPILink Lite and TAPI 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. The Real-time event stream is enabled by installing the CTI Pro licence key onto the system unit.

---

## 1.1 DevLink DLL

The DevLink DLL, which is included on the User CD-ROM should be installed with the licence key. DevLink 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 DevLink 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

- Any application that uses the DevLink 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.
- When developing an application that uses the DLL, the DevLink 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.

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

DevLink 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 DevLink 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.



## 1.2 Connecting to IP Office using DevLink

DevLink 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 DevLink 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

- If DevLink 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.
- 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.
- The pbx\_password parameter should be the monitor password of the switch, not the system password.
- The reserved1 and reserved2 parameters are for future expansion, and should be set to NULL (nil in Delphi).

---

## 1.3 Example: Connecting 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.
```

## 1.4 Example: Connecting 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;
}
```

---

## 1.5 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 );
```

# Chapter 2.

# DevLink Real-Time Event Stream

---

## 2. 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 voicemail system or parked).

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

- [S events](#)<sup>[15]</sup>  
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](#)<sup>[18]</sup>  
D events are generated when the call is completed or abandoned. They indicate that the call no longer exists.
- [A events](#)<sup>[18]</sup>  
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 DevLink 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.

## 2.1 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 call 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: <ul style="list-style-type: none"> <li>• 0 Idle</li> <li>• 1 Ringing</li> <li>• 2 Connected</li> <li>• 3 Disconnected</li> <li>• 4 Suspending</li> <li>• 5 Suspended</li> <li>• 6 Resuming</li> <li>• 7 Dialling</li> <li>• 8 Dialed</li> <li>• 9 Local Dial</li> <li>• 10 Queued</li> <li>• 11 Parked</li> <li>• 12 Held</li> <li>• 13 Redialling</li> </ul>   |  |              |      |            |   |   |
| 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 dial tone)  |  |              |      |            |   |   |
| 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:<br>For extensions: Name (Number)<br>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<br>Achannel   | String  | Aslot is the slot number of the A side of the call and is the identify of the TDM trunk.<br>Achannel is the channel number within the Aslot.<br>The format of the string is "Number.Number".   |  |              |      |            |   |   |
| 13         | Bslot<br>Bchannel   | String  | Bslot is the slot number of the B side of the call and is the identify of the TDM trunk.<br>Bchannel is the channel number within the Bslot.<br>The format of the string is "Number.Number".   |  |              |      |            |   |   |
| 14         | Called party<br>presentation & type   | 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 found in the called party number field.<br>The format of the string is "Number.Number". <table border="1" data-bbox="531 1803 1439 2063"> <thead> <tr> <th></th> <th>Presentation</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>ETSI Types</td> <td> <ul style="list-style-type: none"> <li>• 0 = Presentation allowed.</li> <li>• 1 = Presentation restricted (ie. withhold CLI).</li> <li>• 2 = Presentation not available (ie. call on trunk with no CLI).</li> </ul> </td> <td> <ul style="list-style-type: none"> <li>• 0 = Unknown</li> <li>• 1 = International.</li> <li>• 2 = National.</li> <li>• 3 = Network Specific.</li> <li>• 4 = Subscriber number.</li> </ul> </td> </tr> </tbody> </table> |  | Presentation | Type | ETSI Types | <ul style="list-style-type: none"> <li>• 0 = Presentation allowed.</li> <li>• 1 = Presentation restricted (ie. withhold CLI).</li> <li>• 2 = Presentation not available (ie. call on trunk with no CLI).</li> </ul> | <ul style="list-style-type: none"> <li>• 0 = Unknown</li> <li>• 1 = International.</li> <li>• 2 = National.</li> <li>• 3 = Network Specific.</li> <li>• 4 = Subscriber number.</li> </ul> |
|            | Presentation  | Type  |  |  |              |      |            |   |   |
| ETSI Types | <ul style="list-style-type: none"> <li>• 0 = Presentation allowed.</li> <li>• 1 = Presentation restricted (ie. withhold CLI).</li> <li>• 2 = Presentation not available (ie. call on trunk with no CLI).</li> </ul> | <ul style="list-style-type: none"> <li>• 0 = Unknown</li> <li>• 1 = International.</li> <li>• 2 = National.</li> <li>• 3 = Network Specific.</li> <li>• 4 = Subscriber number.</li> </ul> |  |  |              |      |            |   |   |

| Field                      | Name  | Type  | Description  |                            |   |   |
|----------------------------|---|---|--|----------------------------|---|---|
|                            |   |   | <table border="1"> <tr> <td>Additional IP Office Types</td> <td> <ul style="list-style-type: none"> <li>• 100 = Default</li> </ul> </td> <td> <ul style="list-style-type: none"> <li>• 100 = Default.</li> <li>• 102 = Voicemail call.</li> <li>• 104 = Paging call.</li> </ul> </td> </tr> </table> | Additional IP Office Types | <ul style="list-style-type: none"> <li>• 100 = Default</li> </ul> | <ul style="list-style-type: none"> <li>• 100 = Default.</li> <li>• 102 = Voicemail call.</li> <li>• 104 = Paging call.</li> </ul> |
| Additional IP Office Types | <ul style="list-style-type: none"> <li>• 100 = Default</li> </ul> | <ul style="list-style-type: none"> <li>• 100 = Default.</li> <li>• 102 = Voicemail call.</li> <li>• 104 = Paging call.</li> </ul> |  |                            |   |   |
| 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.<br>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, Voicemail 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 used  | Number  | Non-zero if the service quota is 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  |

---

## 2.2 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.                    |

## 2.3 A events

A events indicate that the call ends have been swapped. This occurs, for example, when the originating extension un parks 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                    |

# Chapter 3.

# Functions

---

## 3. Functions

### 3.1 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 DevLink.
- **pbx\_address**  
The IP address of the IP Office system (either an IP address or a host name can be used). This may be set to "255.255.255.255" in which case DevLink 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;
```

## 3.2 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 DevLink 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;
```

## 3.3 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 DevLink 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;
```



# Chapter 4.

# Callbacks

---

## 4. Callbacks

### 4.1 COMMSEVENT

The COMMSEVENT callback is called by DevLink 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 DevLink. This can occur either because the IP Office system unit is under heavy load, or because the application using DevLink 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 );
```



---

## 4.2 CALLLOGEVENT

The CALLLOGEVENT callback is called by DevLink 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 );
```



# Chapter 5.

# Appendices

---

## 5. Appendices

### 5.1 Appendix B: 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.
```

## 5.2 Appendix A: 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 parml
);
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

## A

A events 14, 18

## C

call log event 25

commsevent 24

Connecting 9

## D

D events 14, 18

devlink

.h 29

.pas 28

DLL 8

real-time event stream 14

disconnecting 12

dlclose 21

dlopen 20

dlregistertype2calldeltas 21

## E

examples 10, 11

## S

S events 14, 15







Performance figures and data quoted in this document are typical, and must be specifically confirmed in writing by Avaya before they become applicable to any particular order or contract. The company reserves the right to make alterations or amendments to the detailed specifications at its discretion. The publication of information in this document does not imply freedom from patent or other protective rights of Avaya or others.

All trademarks identified by the ® or ™ are registered trademarks or trademarks, respectively, of Avaya Inc. All other trademarks are the property of their respective owners.

This document contains proprietary information of Avaya and is not to be disclosed or used except in accordance with applicable agreements.

© 2010 Avaya Inc. All rights reserved.