



# CHAPTER 8

## Cisco Unified TAPI Examples

---

This chapter provides examples that illustrate how to use the Cisco Unified TAPI implementation. This chapter includes the following subroutines:

- [MakeCall](#)
- [OpenLine](#)
- [CloseLine](#)

### MakeCall

```
STDMETHODIMP CTACTrl::MakeCall(BSTR destNumber, long pMakeCallReqID, long hLine, BSTR user2user, long
translateAddr) {
    AFX_MANAGE_STATE(AfxGetStaticModuleState())

    USES_CONVERSION;
    tracer->tracef(SDI_LEVEL_ENTRY_EXIT, "CTACTrl::Makecall %s %d %d %s %d\n",
        T2A((LPTSTR)destNumber), pMakeCallReqID, hLine, T2A((LPTSTR)user2user),
        translateAddr);

    //CtPhoneNo m_pno;
    CtTranslateOutput to;

    //LPCSTR pszTranslatable;
    CString sDialable;

    CString theDestNumber(destNumber);

    CtCall* pCall;
    CtLine* pLine=CtLine::FromHandle((HLINE)hLine);

    if (pLine==NULL) {
        tracer->tracef(SDI_LEVEL_ERROR, "CTACTrl::MakeCall : pLine == NULL\n");
        return S_FALSE;
    } else {
        pCall=new CtCall(pLine);
        pCall->AddSink(this);

        sDialable = theDestNumber;

        if (translateAddr) {
            //m_pno.SetWholePhoneNo((LPCSTR)theDestNumber);
            //pszTranslatable = m_pno.GetTranslatable();
            if (TSUCCEEDED(to.TranslateAddress(pCall->GetLine()->GetDeviceID(),
                (LPCSTR)theDestNumber)) ) {
```

```

        sDialable = to.GetDialableString();
    }
}
TRESULT tr = pCall->MakeCall((LPCSTR)sDialable, 0, this);
if( TPENDING(tr) || TSUCCEEDED(tr) ) {
    //GCGC the correct hCall pointer is not being returned yet
    if (translateAddr)
        Fire_MakecallReply(hLine, (long)tr, (long)pCall->GetHandle(),
            sDialable.AllocSysString());
    else
        Fire_MakecallReply(hLine, (long)tr, (long)pCall->GetHandle(),destNumber);

    return S_OK;
} else {
    //GCGC delete the call that was created above.
    tracer->tracef(SDI_LEVEL_ERROR, "CTActrl::MakeCall : pCall->MakeCall failed\n");
    delete pCall;
    return S_FALSE;
}
}
}
}

```

## OpenLine

```

STDMETHODIMP CTActrl::OpenLine(long lDeviceID, BSTR lineDirNumber, long lPriviledges,
                                long lMediaModes, BSTR receiveIPAddress, long lreceivePort) {
    USES_CONVERSION;
    tracer->tracef(SDI_LEVEL_ENTRY_EXIT, "CTActrl::OpenLine %d %s %d %d %s %d\n",
        lDeviceID, T2A((LPTSTR)lineDirNumber), lPriviledges, lMediaModes,
        T2A((LPTSTR)receiveIPAddress), lreceivePort);

    int lineID;
    TRESULT tr;
    CString strReceiveIP(receiveIPAddress);
    CString strReqAddress(lineDirNumber);

    //bool bTermMedia=((!strReceiveIP.IsEmpty()) && (lreceivePort!=0));
    bool bTermMedia=((lMediaModes & LINEMEDIAMODE_AUTOMATEDVOICE) != 0) &&
        (lreceivePort!=0) && (!strReceiveIP.IsEmpty());
    CtLine* pLine;

    AFX_MANAGE_STATE(AfxGetStaticModuleState())

    tracer->tracef(SDI_LEVEL_DETAILED, "TAC: --> OpenLine()\n");

    if ((lDeviceID<0) && !strcmp((char *)lineDirNumber, "")) {
        tracer->tracef(SDI_LEVEL_ERROR, "TCD: error - bad device ID and no dirn to open\n");
        return S_FALSE;
    }
    lineID=lDeviceID;

    if (lDeviceID<0) {
        //search for line ID in list of lines.
        CtLineDevCaps ldc;
        int numLines=:TfxGetNumLines();
        for( DWORD nLineID = 0; (int)nLineID < numLines; nLineID++ ) {
            if( /*ShouldShowLine(nLineID) &&*/ TSUCCEEDED(ldc.GetDevCaps(nLineID)) ) {
                CtAddressCaps ac;
                tracer->tracef(SDI_LEVEL_DETAILED, "CTActrl::OpenLine :
                    Calling ac.GetAddressCaps %d 0\n", nLineID);
                if ( TSUCCEEDED(ac.GetAddressCaps(nLineID, 0)) ) {

```

```

        // GCGC only one address supported
        CString strCurrAddress(ac.GetAddress());
        if (strReqAddress==strCurrAddress) {
            lineID=nLineID;
            break;
        }
    }

    } else {
        tracer->tracef(SDI_LEVEL_ERROR, "TAC: error - GetAddressCaps() failed\n");
    }
}

if (lDeviceID<0) {
    tracer->tracef(SDI_LEVEL_ERROR,
        "TAC: error - could not find dirn %s to open line.\n", (LPCSTR)lineDirNumber);
    return S_FALSE;
}

// if we are to do media termination; negotiate the extensions version

DWORD retExtVersion;
if (bTermMedia) {
    TRESULT tr3;
    tracer->tracef(SDI_LEVEL_DETAILED,
        "TAC: lineNegotiateExtVersion - appHandle = %d, deviceID = %d, API ver = %d,
        HiVer = %d, LoVer = %d\n", CtLine::GetAppHandle(), lineID,
        CtLine::GetApiVersion(lineID),
        0x80000000 | 0x00010000L,
        0x80000000 | 0x00020000L );
    tr3=::lineNegotiateExtVersion(CtLine::GetAppHandle(),
        lineID, CtLine::GetApiVersion(lineID),
        0x80000000 | 0x00010000L, // TAPI v1.3,
        0x80000000 | 0x00020000L,
        &retExtVersion);
    tracer->tracef(SDI_LEVEL_DETAILED,
        "TAC: lineNegotiateExtVersion returned: %d\n", tr3);
}

pLine=new CtLine();
tr=pLine->Open(lineID, this, lPriviledges, lMediaModes);
if( TSUCCEEDED(tr) ) {
    if (bTermMedia) {
        if (retExtVersion==0x10000) {
            CiscoLineDevSpecificUserControlRTPStream dsucr;
            dsucr.m_RecievePort=lreceivePort;
            dsucr.m_RecieveIP::inet_addr((LPCSTR)strReceiveIP);
            TRESULT tr2;

            tr2=::lineDevSpecific(pLine->GetHandle(),
                0,0, dsucr.lpParams(),dsucr.dwSize());
            tracer->tracef(SDI_LEVEL_DETAILED,
                "TAC: lineDevSpecific returned: %d\n", tr2);
        } else {
            //GCGC here put in the new calls to set the media types!
            CiscoLineDevSpecificUserControlRTPStream2 dsucr;
            dsucr.m_RecievePort=lreceivePort;
            dsucr.m_RecieveIP::inet_addr((LPCSTR)strReceiveIP);
            dsucr.m_MediaCapCount=4;

            dsucr.m_MediaCaps[0].MediaPayload=4;
            dsucr.m_MediaCaps[0].MaxFramesPerPacket=30;
            dsucr.m_MediaCaps[0].G723BitRate=0;
        }
    }
}

```

```

dsucr.m_MediaCaps[1].MediaPayload=9;
dsucr.m_MediaCaps[1].MaxFramesPerPacket=90;
dsucr.m_MediaCaps[1].G723BitRate=1;
dsucr.m_MediaCaps[2].MediaPayload=9;
dsucr.m_MediaCaps[2].MaxFramesPerPacket=90;
dsucr.m_MediaCaps[2].G723BitRate=2;
dsucr.m_MediaCaps[3].MediaPayload=11;
dsucr.m_MediaCaps[3].MaxFramesPerPacket=90;
dsucr.m_MediaCaps[3].G723BitRate=0;

TRESULT tr2;

tr2::lineDevSpecific(pLine->GetHandle(),
                    0,0, dsucr.lpParams(),dsucr.dwSize());
tracer->tracef(SDI_LEVEL_DETAILED,
              "TAC: lineDevSpecific returned: %d\n", tr2);
}
}

CtAddressCaps ac;
LPCSTR pszAddressName;
if ( TSUCCEDED(ac.GetAddressCaps(lineID, 0)) ) {
    // GCGC only one address supported
    pszAddressName = ac.GetAddress();
} else {
    pszAddressName = NULL;
    tracer->tracef(SDI_LEVEL_ERROR, "TAC: error - GetAddressCaps() failed.\n");
}

OpenedLine((long)pLine->GetHandle(), pszAddressName, 0);

// now let's try to open the associated phone device
// Get the phone from the line

DWORDnPhoneID;
bool b_phoneFound=false;
CtDeviceID did;
if((m_bUsesPhones) && TSUCCEDED(did.GetID("tapi/phone", pLine->GetHandle())) ) {
    nPhoneID = did.GetDeviceID();
    tracer->tracef(SDI_LEVEL_DETAILED,
                  "TAC: Retrieved phone device %d for line.\n",nPhoneID);

    // check to see if phone device is already open

    long hPhone;
    CtPhone* pPhone;
    if (!m_deviceID2phone.Lookup((long)nPhoneID,hPhone)) {
        tracer->tracef(SDI_LEVEL_SIGNIFICANT,
                      "TAC: phone device not found in open list, opening it...\n");
        pPhone=new CtPhone();
        TRESULT tr_phone;
        tr_phone=pPhone->Open(nPhoneID,this);
        if (TSUCCEDED(tr_phone)) {
            ::phoneSetStatusMessages(pPhone->GetHandle(),
                                     PHONESTATE_DISPLAY | PHONESTATE_LAMP |
                                     PHONESTATE_HANDSETHOOKSWITCH | PHONESTATE_HEADSETHOOKSWITCH |
                                     PHONESTATE_REINIT | PHONESTATE_CAPSCHANGE | PHONESTATE_REMOVED,
                                     PHONEBUTTONMODE_KEYPAD | PHONEBUTTONMODE_FEATURE |
                                     PHONEBUTTONMODE_CALL |
                                     PHONEBUTTONMODE_LOCAL | PHONEBUTTONMODE_DISPLAY,
                                     PHONEBUTTONSTATE_UP | PHONEBUTTONSTATE_DOWN);
            m_phone2line.SetAt((long)pPhone->GetHandle(), (long)pLine->GetHandle());
            m_line2phone.SetAt((long)pLine->GetHandle(), (long)pPhone->GetHandle());
            m_deviceID2phone.SetAt((long)nPhoneID, (long)pPhone->GetHandle());
        }
    }
}

```

```

        m_phoneUseCount.SetAt((long)pPhone->GetHandle(), 1);
    } else {
        tracer->tracef(SDI_LEVEL_ERROR,
            "TAC: error - phoneOpen failed with code %d\n", tr_phone);
    }
} else {
    pPhone=CtPhone::FromHandle((HPHONE)hPhone);
    long theCount;

    if (m_phoneUseCount.Lookup((long)pPhone->GetHandle(), theCount))
        m_phoneUseCount.SetAt((long)pPhone->GetHandle(), theCount+1);
    else {
        //GCGC this would be an error condition!
        tracer->tracef(SDI_LEVEL_ERROR,
            "TAC: error - m_phoneUseCount does not contain phone entry.\n");
    }
}
} else {
    tracer->tracef(SDI_LEVEL_ERROR,
        "TAC: error - could not retrieve PhoneID for line.\n");
}
tracer->tracef(SDI_LEVEL_DETAILED, "TAC: <-- OpenLine()\n");
return S_OK;
} else {
    tracer->tracef(SDI_LEVEL_ERROR, "TAC: error - lineOpen failed: %d\n", tr);
    tracer->tracef(SDI_LEVEL_DETAILED, "TAC: <-- OpenLine()\n");
    OpenLineFailed(tr,0);
    delete pLine;
    return S_FALSE;
}
}
}

```

## CloseLine

```

STDMETHODIMP CTActrl::CloseLine(long hLine) {

    AFX_MANAGE_STATE(AfxGetStaticModuleState())

    tracer->tracef(SDI_LEVEL_ENTRY_EXIT, "CTActrl::CloseLine %d\n", hLine);

    CtLine* pLine;
    pLine=CtLine::FromHandle((HLINE) hLine);

    if (pLine!=NULL) {
        // close the line
        pLine->Close();
        // remove it from the list
        delete pLine;
        long hPhone;
        long theCount;
        if ((m_bUsesPhones) && (m_line2phone.Lookup(hLine,hPhone))) {
            CtPhone* pPhone=CtPhone::FromHandle((HPHONE)hPhone);
            if (pPhone!=NULL) {
                if (m_phoneUseCount.Lookup(hPhone,theCount))
                    if (theCount>1) {
                        // decrease the number of lines using this phone
                        m_phoneUseCount.SetAt(hPhone,theCount-1);
                    }
                else {
                    //nobody else is using this phone, so let's remove it.
                    m_deviceID2phone.RemoveKey((long)pPhone->GetDeviceID());
                }
            }
        }
    }
}

```

```
        m_phone2line.RemoveKey(hPhone);
        m_phoneUseCount.RemoveKey(hPhone);

        //now let's close the phone
        pPhone->Close();
    }
    //either way, remove the map entry from line to phone.
    m_line2phone.RemoveKey(hLine);
}
return S_OK;
}
else
return S_FALSE;
}
```