

How_to_subscribe_presence_status_using_SIP_APIs

This sample code is used for subscribing the presence status like online/offline. for event subscription we use CSIPSubscribeDialogAssoc class for sending subscribe request. and for event we use eventheader classes.

code snippet

```
CMyExCode::SubscribeEvent ()
{
    CSIPMessageElements* msgElem = CSIPMessageElements::NewL();
    CleanupStack::PushL( msgElem );

    _LIT8(KMediaType, "application");
    _LIT8(KMediaSubType, "pidf+xml");
    CSIPContentTypeHeader* ct = CSIPContentTypeHeader::NewLC(KMediaType, KMediaSubType);

    TBuf8<512> doc;

    _LIT8(kdoc1, "<?xml version=\"1.0\" encoding=\"UTF-8\"?><presence
        xmlns=\"urn:ietf:params:xml:ns:pidf\" entity=\"\"");
    _LIT8(kdoc2, "\"><tuple id=\"tup1\"
    ><status><basic>open</basic></status><note>");
    _LIT8(kdoc3, "</note></tuple></presence>"); //presence document

    doc.Copy(kdoc1);
    doc.Append(*aor);
    doc.Append(kdoc2);
    doc.Append(_L8("Online"));
    doc.Append(kdoc3);

    msgElem->SetContentL(doc.AllocL(), ct);
    CleanupStack::Pop( ct );

    //Get the current connection
    CSIPConnection& conn = ConnectionL();

    // Create user header array
    RPointerArray<CSIPHeaderBase>* userHeadersArr = new(ELeave)RPointerArray<CSIPHeaderBase>;
    CleanupStack::PushL( userHeadersArr );

    RDebug::Print(_L("Start Of SubscribeL4"));

    // Create Expires Header
    CSIPExpiresHeader* expiresHdr = new (ELeave)CSIPExpiresHeader(60);

    CleanupStack::PushL( expiresHdr );

    userHeadersArr->Append(expiresHdr);
    CleanupStack::Pop( expiresHdr );

    // Set the User Headers
    msgElem->SetUserHeadersL(*userHeadersArr);
    CleanupStack::Pop( userHeadersArr );

    //Create Event Header
    CSIPEventHeader* eventHdr = CSIPEventHeader::NewL(_L8("PRESENCE"));
    CleanupStack::PushL( eventHdr );
    userHeadersArr->Append(eventHdr);
```

How_to_subscribe_presence_status_using_SIP_APIs

```
// create an object for refresh
CSIPRefresh* iRefresh = CSIPRefresh::NewL();
CleanupStack::PushL( iRefresh );

// create an object for CSIPSubscribeAssoc

iSIPSubscribeDialogAssoc = CSIPSubscribeDialogAssoc::NewL
( conn, fromHeader, uri8, eventHdr, toHeader, cont);

CleanupStack::PushL( iSIPSubscribeDialogAssoc );

CSIPClientTransaction* clientTx ;
TRAPD(err, clientTx = iSIPSubscribeDialogAssoc->SendSubscribeL
(msgElem, iRefresh));

if(err!=KErrNone)
{
    User::Leave(err);
}

CleanupStack::Pop( iSIPSubscribeDialogAssoc );
CleanupStack::Pop( iRefresh );

CleanupStack::Pop( eventHdr );
CleanupStack::Pop( msgElem );
delete clientTx;
}
```