SMS Utilities API

SMS Utilities API provides methods for sending the sms using sockets.

Note:

- This API is not part of the public SDK. It can be found in the SDK API Plug-in.
- NB! If another application has already opened the socket then this method fails!

Use cases

Sending and receiving SMS silently (that is without the message tone and without any new message notification) can be achieved by sending SMS through sockets and also by listening for incoming SMS with the help of sockets.

Example code

SMSSendL() method is responsible for sending the sms using sockets.

```
void SMSSendL()
     {
     //iFileServer is RFs object
     iFileSession.Connect();
     //iSockerServer is RSocketServ object
     iSocketServer.Connect();
     //iSocket is RSocket Object
     TInt err1 = iSocket.Open(iSocketServer,KSMSAddrFamily,
               KSockDatagram, KSMSDatagramProtocol);
     if(!err1)
     {
      //SMS address for a socket.
     TSmsAddr smsaddr;
     //Only for sending, no reception.
     smsaddr.SetSmsAddrFamily(ESmsAddrSendOnly);
     TInt BindErr1= iSocket.Bind(smsaddr);
     if(BindErr1 == KErrNone)
     {
     CSmsBuffer *buffer = CSmsBuffer::NewL();
     CleanupStack::PushL(buffer);
     //Inserting msg. to be sent to a buffer
     buffer->InsertL(0, _L("MyMessage"));
     //Stream that writes a CSmsMessage object across a socket
     RSmsSocketWriteStream writestream(iSocket);
     //ESmsSubmit-SMS-SUBMIT, sent from Mobile Station to Service Center
     iSmsMessage = CSmsMessage::NewL(iFileSession,CSmsPDU::ESmsSubmit,
           buffer);
     //Sets the message Service Center Address via which msg.
     //will be sent to receipent
     iSmsMessage->SmsPDU().SetServiceCenterAddressL(_L("+01234567890"));
     //Sets destination number
     iSmsMessage->SmsPDU().SetToFromAddressL(_L("+09876543210"));
     //Externalises message to a stream which is used for
     //writing data into the socket
     iSmsMessage->ExternalizeL(writestream);
     //Ensures that any buffered data is written to the stream.
     writestream.CommitL();
     TPckgBuf<TInt> sendBuffer;
     sendBuffer=KSockSelectWrite;
     //Applies an asynchronous I/O control operation on a socket.
     iSocket.Ioctl(KIoctlSendSmsMessage,iStatus,&sendBuffer,KSolSmsProv);
     iRead=EFalse;
     SetActive();
     CleanupStack::PopAndDestroy(buffer);
     }
     }
     }
```

SMSRead() is responsible for Reading incoming SMS silently.Actually it checks whether the incoming message is of a specific patern(In this case the message should start with "##")

```
void SMSRead()
     {
     TBuf8<2> matchTag;
     _LIT8(KTag1,"##");
     matchTag.Copy(KTag);
     iReadServer.Connect();
     //Opens a socket by creating a new subsession to the socket server.
     TInt err = iReadSocket.Open(iReadServer,KSMSAddrFamily,
                    KSockDatagram,  KSMSDatagramProtocol);
     if(!err)
 {
 TSmsAddr smsAddr;
 // App. listens for sms msgs with some special tag in it.
   smsAddr.SetSmsAddrFamily(ESmsAddrMatchText);
 smsAddr.SetTextMatch(KTag1);
 TInt bindErr = iReadSocket.Bind(smsAddr);
 if(!bindErr)
   {
   sbuf() = KSockSelectRead;
   //Applies an asynchronous I/O control operation on a socket.
   iReadSocket.Ioctl( KIOctlSelect,iStatus, &sbuf, KSOLSocket);
   iRead=ETrue;
   SetActive();
   }
 }
```


(C) Copyright Nokia 2013. All rights reserved.

```
    }
```

```
void RunL()
     {
     if(iRead)
 {
        iFileSession.Connect()
 TBuf<2> matchTag;
 matchTag.Copy(KTag);
 CSmsBuffer *buffer=CSmsBuffer::NewL();
        CleanupStack::PushL(buffer);
 //Stream that reads a CSmsMessage object across a socket.
 RSmsSocketReadStream readStream(socket1);
 //Allocates and creates a CSmsMessage
 //ESmsDeliver-SMS-DELIVER, sent from service center to Station.
        CSmsMessage message = CSmsMessage::NewL
                        TheFs1,CSmsPDU::ESmsDeliver,buffer);
        CleanupStack::PushL(message);

        //Internalises data from stream to CSmsMessage
 message->InternalizeL(readStream);
 readStream.Close();
 //Extracting the received message to a buffer
        TBuf<255> msgContents;
 message->Buffer().Extract(msgContents,0,message->Buffer().Length());
        CleanupStack::PopAndDestroy(2)
        //compare whether the incoming message starts with "##"
 if( (buf1.Left(2)).Compare(matchTag) == 0)
   {
   iReadSocket.Ioctl( KIoctlReadMessageSucceeded,iStatus, &sbuf,KSolSmsProv);
   //Now msgContents contains the actual message.
           iRead=EFalse;
   SetActive();
   }
        }
     }
```

Note: sBuf is of type TPckgBuf<TUint>

Example Application

File:SilentSMS.zip

See Also

See RSendAs class for sending SMS and CMsvSession and MMsvSessionObserver::HandleSessionEventL() for receiving SMS with public APIs. This approach may play the incoming message tone.

Related Links:

- Sending SMS with RSendAs
- SMS Receiver
- Reading SMS from Inbox
- Sending SMS in S60 3rd Edition - MTM
- SMS Operations
- How to send an SMS using sockets
- Sending-Receiving SMS through an Exe (Server)
- Create Local SMS
- How to Open SMS or MMS Editor

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.