SMS operations using Symbian C++

This article illustrates various operations performed on SMS including sending messages, reading incoming messages and message folders, deleting messages, stopping outgoing SMS from being sent, disabling delivery reports and sending messages to multiple recipients.

Prerequisite

31 May 2009

- Download **SmsHandler.zip** as follows:
  - Media:SMShandler_S60_2nd.zip (for S60 2nd edition only)
  - Media:SMShandler_S60_3rd.zip (for S60 3rd edition only)
- Extracting **SmsHandler.zip** will result into **SmsHandler.h** and **SmsHandler.cpp**
- Copy-Paste **SmsHandler.h** into your project's **/inc** folder.
- Copy-Paste **SmsHandler.cpp** into your project's **/src** folder.
- Edit your **.mmp** file. Add an entry for **SmsHandler.cpp** in **SOURCE** directive.

```
SOURCE SMSHandler.cpp
```

- Edit your **.mmp** file. Add libraries for SMS handling, and capabilities required.

```
//Libraries included for SMS support-
LIBRARY msgs.lib smcm.lib gsmu.lib mtur.lib
CAPABILITY ReadUserData WriteUserData NetworkServices
```

- Open your **CYrApplicationContainer.h** file.
- Include **SmsHandler.h**.

```
#include "SMSHandler.h" //Added for SMS Handling
```

- Define an object of SmsHandler class.

```
    private: //data
        ...
 CSmsHandler* iSmsHandler;
```

- Open your **CYrApplicationContainer.cpp** file.
- Initialize SmsHandler object.

```
CYrApplicationContainer::ConstructL()
{
...
    SetRect(aRect);
    ActivateL();

    iSmsHandler = CSmsHandler::NewL(); // SmsHandler
}
```

Send message

- Define one function of your own SendMsg():

```
void CYrApplicationContainer ::SendMsg()
{
    TBuf<128> SMSText,PhoneNumber;
    SMSText.Copy(_L("Test Message"));
    PhoneNumber.Copy(_L("999999999")); //Replace Number as per your needs

    iSmsHandler->SendL( PhoneNumber, SMSText) ;
}
```

Read Messaging folders

The messaging folders are inbox, outbox, draft and set items. The following code snippet illustrates reading messages from **Inbox Folder** using `KMsvGlobalInBoxIndexEntryId`. The same approach can be used for the other folders using the appropriate id: `KMsvGlobalOutBoxIndexEntryId`, `KMsvDraftEntryId`, `KMsvSentEntryId`

```
void CSmsHandler::ReadInbox()
{
        HBufC* SMSContent = HBufC::NewLC(400);
 HBufC8* SMSContent8 = HBufC8::NewLC(400);

 TMsvSelectionOrdering sort;
 sort.SetShowInvisibleEntries(ETrue); // we want to handle also the invisible entries

 CMsvEntry* inboxContext=CMsvEntry::NewL(*iSession,KMsvGlobalInBoxIndexEntryId,sort); // Reading Messages from Inbox Folder
```


(C) Copyright Nokia 2013. All rights reserved.

```
 CleanupStack::PushL(inboxContext);

 CMsvEntrySelection* entries = inboxContext->ChildrenL();
 CleanupStack::PushL( entries );

 TInt msgCount= entries->Count();
 for (TInt i=0; i<entries->Count(); i++)
 {

  TMsvId entryID = entries->At(i);
  iSmsMtm->SwitchCurrentEntryL(entryID);

  CMsvEntry* entry= iSession->GetEntryL((*entries)[i]);
  CleanupStack::PushL(entry);

  CMsvStore* inboxStore;
                /* Skip loop if fail in obtains the message store */
  TRAPD(r, inboxStore = entry->ReadStoreL());

  if(KErrNone != r)
  {
   CleanupStack::PopAndDestroy(entry);
   continue;
  }

  CleanupStack::PushL(inboxStore);
  if (inboxStore->HasBodyTextL())
  {
                        TMsvEntry entry1 =  entry->Entry();
                        /* Gives you phone Number or Contact Name if Contact is present*/
   TBufC<50>text(entry1.iDetails);
                        TBuf16<30> msg;
   msg.Copy(_L("Name: "));
   msg.Append(text);

                         /*If you want to get the Recipient Number, when iDetails gives you Contact's Name. */
                        iSmsMtm->LoadMessageL();
                        CSmsHeader& header = iSmsMtm->SmsHeader();
                        /* This will give you actual phone number irrespective of the name of contact*/
                        TPtrC from = header.FromAddress();
                        TBuf8<60> number;
   number.Copy(_L("Number: "));
   number.Append(from);

                       /* To read phone number from Sent Item folder, see the Note below */

   CRichText& richText= iSmsMtm->Body();
   inboxStore->RestoreBodyTextL(richText);
   const TInt length = richText.DocumentLength();

                        /* Gives you actual content(Body) of SMS */
   SMSContent->Des().Copy(richText.Read(0,length));
   richText.Reset();

   SMSContent8->Des().Copy(SMSContent->Des());

                         /* Write SMS Body in the SMSBody.txt file*/
   //WriteToFile(SMSContent8->Des());

                        /* Write SMS Body, number and contact name in file*/
                        WriteToFile(msg, number,SMSContent8->Des());
  }
  else
  {
  // no text in SMS
  }
  CleanupStack::PopAndDestroy(2,entry);
 }
 CleanupStack::PopAndDestroy(4,SMSContent);
}
```

To read phone number from the message in the Sent Item folder, use the following code snippet.

```
CSmsHeader& smsHeader = iSmsMtm->SmsHeader();
const CArrayPtrFlat<CSmsNumber>&  array= smsHeader.Recipients();
CSmsNumber* smsNumber = array.At(0);
TPtrC recipentNumber = smsNumber->Address(); // recipentNumber will contain the recipient Number
```

```
void CSmsHandler::WriteToFile(const TDesC8& aSMSContent8)
{
 _LIT(KFileSpec,"C:\\data\\SMSBody.txt");//File, in which SMS Body will be stored
 TInt pos = 0;
 RFs fs;
 fs.Connect();
 RFile file;
 TInt err = file.Open(fs, KFileSpec, EFileWrite);

 if (err == KErrNotFound)// file does not exist - create it
 {
     err = file.Create(fs, KFileSpec, EFileWrite);
 }
 if(err == KErrNone)
 {
  file.Seek(ESeekEnd, pos);
  file.Write(aSMSContent8);
  file.Close();
 }
 fs.Close();
}

void CSmsHandler::WriteToFile(const TDesC8& aMsgHeader, const TDesC8& aNumber,const TDesC8& aMsgBody)
{
 RFile file;
 /* Open a file if it exists, otherwise create new one*/
 if (KErrNone != file.Open(CCoeEnv::Static()->FsSession(), _L("c:\\Data\\SmsBackup.txt"), EFileWrite))
  file.Replace(CCoeEnv::Static()->FsSession(), _L("c:\\Data\\SmsBackup.txt"), EFileWrite);
 TInt pos = 0;
 file.Seek(ESeekEnd, pos);
 file.Write(_L8("\r\n"));
 /*Write contact name, if available*/
 file.Write(aMsgHeader);
 file.Write(_L8("\n"));
 /*Write mobile number*/
 file.Write(aNumber);
 file.Write(_L8("\n"));
```


(C) Copyright Nokia 2013. All rights reserved.

```
 /*Write message*/
 file.Write(aMsgBody);
 /*Line to separate two messages*/
 file.Write(_L8("\n-------------------------"));
 file.Close();
}
```

Read incoming message (Online Mode)

You will find `MessageReceivedL()` in the **SMSHandler.cpp** file. Perform following changes to read incoming message body and phone number.

```
void CSmsHandler::MessageReceivedL( TMsvId aEntryId )
{
 CMsvEntry* serverEntry = iSession->GetEntryL( aEntryId );   // current entry
 CleanupStack::PushL( serverEntry );
 TMsvEntry entry = serverEntry->Entry(); // currently handled message entry

 entry.SetNew( ETrue );
 entry.SetUnread( ETrue );
 entry.SetVisible( ETrue );

 serverEntry->ChangeL( entry );  // commit changes

 //Added to retrieve message body
 const TDesC& descp = entry.iDescription; // iDescription will have only first 32 characters from the message
 TBuf8<40> MessageArrived;
 MessageArrived.Copy(descp);

 //Added to retrieve Phone Number of the Sender
 iSmsMtm->SwitchCurrentEntryL(aEntryId);
 iSmsMtm->LoadMessageL();
 CSmsHeader& header = iSmsMtm->SmsHeader();

 TPtrC from = header.FromAddress();
 const TDesC& phoneNumber = from;

 CleanupStack::PopAndDestroy( serverEntry );
}
```

Delete message from Sent Item (Online Mode)

Add the following case in the `HandleSessionEventL()` of **SmsHandler.cpp**. So when you send message from your application, it will be deleted from "Sent Item" folder.

```
case EMsvEntriesMoved:
{
 // Entry id is obtained from the session event arguments.
            TMsvId* entryId = STATIC_CAST( TMsvId*, aArg2 );

 // We are interested in messages that are moved to Sent Item Folder
 if ( *entryId == KMsvSentEntryId )
            {
  TMsvSelectionOrdering sort;
               sort.SetSorting(EMsvSortByDateReverse);
  sort.SetShowInvisibleEntries(ETrue); // we want to handle also the invisible entries

  CMsvEntry* parentEntry = CMsvEntry::NewL(*iSession, KMsvSentEntryId, sort);
  CleanupStack::PushL(parentEntry);

  CMsvEntrySelection* entries = parentEntry->ChildrenL();
  CleanupStack::PushL(entries);

  for(TInt i = 0; i < entries->Count(); i++)
  {
    if( parentEntry->ChildDataL(entries->At(i)).iMtmData3 != KUidMsgTypeSMS.iUid )
    {
     parentEntry->DeleteL(entries->At(i));
     break;
    }
  }
  CleanupStack::PopAndDestroy( entries );
  CleanupStack::PopAndDestroy( parentEntry );
  }
  break;
 }
}
```

Delete message from Outbox folder (Online Mode)

Sometimes a need arises to delete message from the outbox folder so that it cannot be scheduled to be sent. The following case can be added in the `HandleSessionEventL()` of **SmsHandler.cpp** in order to delete the message from the Outbox folder when it is sent. A phone number and the message contents can be retrieved before deleting the message which is shown in the following code snippet.

```
case EMsvEntriesMoved:
{
 // Entry id is obtained from the session event arguments.
 TMsvId* entryId = STATIC_CAST( TMsvId*, aArg2 );

 // We are interested in messages that are moved to Sent Item Folder
 if (*entryId == KMsvGlobalOutBoxIndexEntryId)
 {
  TMsvSelectionOrdering sort;
  sort.SetSorting(EMsvSortByDateReverse);
  sort.SetShowInvisibleEntries(ETrue); // we want to handle also the invisible entries
```


(C) Copyright Nokia 2013. All rights reserved.

```
  CMsvEntry* parentEntry = CMsvEntry::NewL(*iSession,
    KMsvGlobalOutBoxIndexEntryId, sort);
  CleanupStack::PushL(parentEntry);

  CMsvEntrySelection* entries = parentEntry->ChildrenL();
  CleanupStack::PushL(entries);

  for (TInt i = 0; i < entries->Count(); i++)
  {
   if (parentEntry->ChildDataL(entries->At(i)).iMtmData3
     != KUidMsgTypeSMS.iUid)
   {
    TMsvId entryID = entries->At(i);
    iSmsMtm->SwitchCurrentEntryL(entryID);
    iSmsMtm->LoadMessageL();

    CSmsHeader& smsHeader = iSmsMtm->SmsHeader();
    const CArrayPtrFlat<CSmsNumber>&  array= smsHeader.Recipients();
    CSmsNumber* smsNumber = array.At(0);
    TPtrC recipentNumber = smsNumber->Address(); // Gives actual phone number, instead of giving mapped name from phonebook

    CRichText& richText= iSmsMtm->Body();
    TInt length = richText.DocumentLength();
    HBufC* SMSContent = HBufC::NewLC(length);
    SMSContent->Des().Copy(richText.Read(0,length)); // Gives you actual content(Body) of SMS
    richText.Reset();
    // Use SMSContent in your function
    CleanupStack::PopAndDestroy(SMSContent);
    parentEntry->DeleteL(entries->At(i));
    break;
   }
  }
  CleanupStack::PopAndDestroy(entries);
  CleanupStack::PopAndDestroy(parentEntry);
 }
 break;
}
```

Note: As the message is already deleted before it gets scheduled for actual sending, the code will leave in the *ScheduleL()* function. To prevent showing the error code on the actual device, use `TRAP_IGNORE` macro around *InvokeAsyncFunctionL* call as shown in the following code.

```
TRAP_IGNORE(iOperation = iSmsMtm->InvokeAsyncFunctionL( ESmsMtmCommandScheduleCopy,
                          *selection, dummyParams, iStatus ));
```

Stop an outgoing SMS

As shown in the following function, you have to filter the specific entry type of the message, hence, you need to get the notification of outgoing message when it is created to be sent to the other party.

```
void CSmsHandler::HandleSessionEventL( TMsvSessionEvent aEvent, TAny* aArg1, TAny* aArg2, TAny* aArg3)
{
...
case EMsvEntriesMoved:
               TMsvId* entryId = STATIC_CAST( TMsvId*, aArg2 );

 // We are interested in messages that are moved to outbox Item Folder
               if ( *entryId == KMsvGlobalOutBoxIndexEntryId  )
                {
     CMsvEntrySelection* newEntries = STATIC_CAST( CMsvEntrySelection*, aArg1 );
                        CMsvEntry* entry= iSession->GetEntryL(newEntries->At(0));
     TMsvEntry e =  entry->Entry();
     e.SetSendingState(KMsvSendStateUnknown);
                        entry->ChangeL(e);
                   }
     break;
...
}
```

As soon as you get the notification for outgoing message being moved to outbox , catch it and change its **SendingState** to *unknown*, the message will not be sent out by MTM server.

Delete messages from Messaging Folders

The following code snippet illustrates deleting messages from **Inbox Folder** - `KMsvGlobalInBoxIndexEntryId` is used. To delete from the outbox, draft or sent item folders use the `KMsvGlobalOutBoxIndexEntryId`, `KMsvDraftEntryId` or `KMsvSentEntryId` respectively.

```
void CSmsHandler::DeleteMessages()
{
 TMsvSelectionOrdering sort;
 sort.SetShowInvisibleEntries(ETrue); // we want to handle also the invisible entries

 CMsvEntry* inboxContext=CMsvEntry::NewL(*iSession,KMsvGlobalInBoxIndexEntryId,sort);
 CleanupStack::PushL(inboxContext);

 CMsvEntrySelection* entries = inboxContext->ChildrenL();
 CleanupStack::PushL( entries );

 TInt msgCount= entries->Count();
 TInt i;
 for (i=0; i < msgCount; i++)
 {
  TMsvId entryID = entries->At(i);
  iSmsMtm->SwitchCurrentEntryL(entryID);

  CMsvEntry* entry= iSession->GetEntryL((*entries)[i]);
  CleanupStack::PushL(entry);

  entry->DeleteL(entryID);
```


(C) Copyright Nokia 2013. All rights reserved.

```
  CleanupStack::PopAndDestroy(entry);
 }
 CleanupStack::PopAndDestroy(entries);
 CleanupStack::PopAndDestroy(inboxContext);
}
```

Disable "Delivery Report"

Open **SmsHandler.cpp** and add {{{1}}} in `CreateMsgL()`

```
TBool CSmsHandler::CreateMsgL()
{
...
 settings->CopyL( iSmsMtm->ServiceSettings() );    // restore settings
     settings->SetDelivery( ESmsDeliveryImmediately ); // to be delivered immediately
 settings->SetDeliveryReport(EFalse);// Delivery Report Disabled here
 header.SetSmsSettingsL( *settings );              // new settings
...
}
```

Receiving and deleting received delivery reports is illustrated in Media:SMS_DeliveryReport_Deleting.zip example.

Send message to multiple recipients

Open **SmsHandler.cpp** and set multiple numbers in `AddAddresseeL()` in `CreateMsgL()`

```
TBool CSmsHandler::CreateMsgL()
{
...
 // Recipient number is displayed also as the recipient alias.
 entry.iDetails.Set( iRecipientNumber );

 // Add addressee.
 TBuf<15> PhoneNumber2;
 PhoneNumber2.Copy(_L("9999999999")); //This is second number on which message will be sent
 iSmsMtm->AddAddresseeL( iRecipientNumber, entry.iDetails );
 iSmsMtm->AddAddresseeL( PhoneNumber2, entry.iDetails );
...
}
```

Related Links

- Sending SMS with RSendAs
- SMS Utilities API
- SMS Receiver
- Reading SMS from Inbox
- Sending SMS in S60 3rd Edition - MTM
- 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.