

ID	TSS001332	Creation date	April 3, 2009
Platform	S60 3rd Edition, FP1 S60 3rd Edition, FP2 S60 5th Edition	Devices	Tested on Nokia N95, Nokia 6210 Navigator, Nokia 5800 XpressMusic
Category	Symbian C++	Subcategory	Messaging

Keywords (APIs, classes, methods, functions):

Overview

This code example can be used to delete the SMS messages stored in the SIM card.

Preconditions

The class should be derived from `MMsvSessionObserver`.

MMP file

```
LIBRARY msgs.lib smcm.lib
```

Header file

```
//includes
#include <msvapi.h>
#include <msvids.h>
#include <MTCLREG.h>
#include <smsclnt.h>
#include <smut.h>
#include <SMUTSET.H>
#include <smcmds.h>

//declarations
CMsvSession* iMsvSession;
CSmsClientMtm* iSmsMtm;
TMsvId iSimFolderId;
CClientMtmRegistry* iClientMtmReg;
```

Source file

```
void CSmsEngine::ConstructL()
{
    iMsvSession = CMsvSession::OpenAsyncL(*this);
    iClientMtmReg = CClientMtmRegistry::NewL(*iMsvSession);
    iSmsMtm = static_cast<CSmsClientMtm*>(iClientMtmReg->NewMtmL (KUidMsgTypeSMS));
}

```

This function reads the SMS messages on the SIM and creates a copy of those messages in an invisible folder under the SMS service in the message store. If successful, the `iEnumerateFolder` member of the operation's progress identifies the invisible folder which contains the messages read from the SIM card.

```
void CSmsEngine::EnumeratePhoneStore()
{
    // Find the service entry
    CMsvEntry* serviceEntry = iMsvSession->GetEntryL(KMsvRootIndexEntryId);
    CleanupStack::PushL(serviceEntry);
    TMsvId serviceId;
    TSmsUtilities::ServiceIdL(*serviceEntry, serviceId, KUidMsgTypeSMS);
    CMsvEntrySelection* selection = new (ELeave) CMsvEntrySelection();
    CleanupStack::PushL(selection);
    selection->AppendL(serviceId);
    TBuf8<1> emp (KNullDesC8);
    CMsvOperationActiveSchedulerWait* waiter = CMsvOperationActiveSchedulerWait::NewLC();
    CMsvOperation* operation = iSmsMtm->InvokeAsyncFunctionL(
        ESmsMtmCommandEnumeratePhoneStores,
        *selection,
        emp,
        waiter->iStatus);
    waiter->Start();
    TSmsProgressBuf progressBuf;
    progressBuf.Copy(operation->ProgressL());
    TSmsProgress progress = progressBuf();
    if (!progress.iError)
    {
        //identify the invisible folder that contains the messages read from the phone store(SIM)
        iSimFolderId = progress.iEnumerateFolder;
    }
    CleanupStack::PopAndDestroy( 2, serviceEntry ); // serviceEntry, selection
    DeleteSimMessages();
}

```

This function actually deletes the specified messages from the SIM card.

```
void CSmsEngine::DeleteSimMessages()
{
    TMsvSelectionOrdering sort;
    sort.SetShowInvisibleEntries(ETrue); // To handle invisible entries also
    CMsvEntry* root = iMsvSession->GetEntryL( KMsvRootIndexEntryId );
    TMsvId iSmsServiceId;
    CleanupStack::PushL( root );
    TSmsUtilities::ServiceIdL( *root, iSmsServiceId );
    CleanupStack::PopAndDestroy( root );
    CMsvEntrySelection* selection = new (ELeave) CMsvEntrySelection();
    selection->AppendL(iSmsServiceId); //The first entry ID in aSelection must be the SMS service ID
    CleanupStack::PushL( selection );
    CMsvEntry* inboxContext=CMsvEntry::NewL(*iMsvSession,iSimFolderId ,sort);
    CleanupStack::PushL(inboxContext);
}

```

TSS001332_-_Deleting_messages_stored_on_SIM_card

```
CMsvEntrySelection* entries = inboxContext->ChildrenL();
CleanupStack::PushL( entries );
TInt count=entries->Count();
while(count)
{
    //All following entry IDs in the selection must then represent each message to be deleted
    selection->AppendL(entries->At(count-1));    count--;
}
TBuf8<1> emp (KNullDesC8);
CMsvOperationActiveSchedulerWait* waiter = CMsvOperationActiveSchedulerWait::NewLC();
iSmsMtm->InvokeAsyncFunctionL(
ESmsMtmCommandDeleteFromPhoneStore,
*selection,
emp,
waiter->iStatus);
waiter->Start();
CleanupStack::PopAndDestroy(2,inboxContext);
CleanupStack::PopAndDestroy(selection);
}
```

Related article: [SMS Operations](#)