Sending an MMS message using Symbian C++

Overview

This snippet demonstrates how to send an MMS message.

MMP file

The following libraries and capabilities are required:

```
CAPABILITY ReadUserData WriteUserData NetworkServices
LIBRARY msgs.lib
```

Header

```
#include <msvapi.h>                         // for MMsvSessionObserver
#include <mmsclient.h>                      // for CMmsClientMtm

// Forward declarations
class CClientMtmRegistry;
class CMsvSession;

class CMMSSender : public CBase, public MMsvSessionObserver
    {
    public:
        void ConstructL();
        virtual ~CMMSSender();

        /*
         * Creates client MTM registry when session is ready for use.
         * This completes model construction and is called after 'server
         * ready' event is received after async opening of CMsvSession.
         */
        void CompleteConstructL();

        // Send MMS message
        void SendMMSL();

    private:
        // from MMsvSessionObserver
        void HandleSessionEventL(TMsvSessionEvent aEvent,
        TAny* aArg1, TAny* aArg2, TAny* aArg3);

    private:
        // Client session on the message server
        CMsvSession*        iSession;

        // Message Type Module (MMS)
        CMmsClientMtm*      iMmsMtm;

        // Mtm client registry for creating new mtms
        CClientMtmRegistry* iMtmReg;

        // CMsvEntry accesses and acts upon a particular Message Server entry
        CMsvEntry*          iMsvEntry;
    };
```

Source

```
#include <mtclreg.h>                // for CClientMtmRegistry
#include <msvids.h>                 // for Message type IDs
#include <CMsvMimeHeaders.h>        // Attachemt mimeheader
#include <eikenv.h>


void CMMSSender::ConstructL()
    {
    // Create CMsvSession
    // Note: New session is opened asynchronously
    iSession = CMsvSession::OpenAsyncL(*this);
    }

void CMMSSender::CompleteConstructL()
    {
    // We get a MtmClientRegistry from our session
    // this registry is used to instantiate new mtms.
    iMtmReg = CClientMtmRegistry::NewL(*iSession);
    iMmsMtm = (CMmsClientMtm*) iMtmReg->NewMtmL( KUidMsgTypeMultimedia );
    }

CMMSSender::~CMMSSender()
    {
```


(C) Copyright Nokia 2013. All rights reserved.

```
    }
    delete iMmsMtm;
    delete iMtmReg;
    delete iMsvEntry;
    delete iSession;
    }

void CMMSSender::HandleSessionEventL(TMsvSessionEvent aEvent,
TAny*, TAny*, TAny*)
    {
    switch (aEvent)
        {
        // This event tells us that the session has been opened
        case EMsvServerReady:
            {
            // Construct the mtm registry & MMS mtm
            CompleteConstructL();
            break;
            }
        default:
            break;
        }
    }

void CMMSSender::SendMMSL()
    {
    // CMsvEntry accesses and acts upon a particular Message Server entry.

    // - NewL() does not create a new entry, but simply a new object to
    //   access an existing entry.

    // - It takes in as parameters the client's message server session,
    //   ID of the entry to access and initial sorting order
    //   of the children of the entry.
    CMsvEntry* entry = CMsvEntry::NewL(*iSession,
    KMsvGlobalOutBoxIndexEntryId ,TMsvSelectionOrdering());
    CleanupStack::PushL(entry);

    // Set context to the parent folder (Outbox)
    iMmsMtm->SwitchCurrentEntryL( entry->EntryId() );

    // Create new message in the parent folder (Outbox) and
    // set it as the current context
    iMmsMtm->CreateMessageL( iMmsMtm->DefaultServiceL()  );
    CleanupStack::PopAndDestroy(); // entry

    // Setting recipients
    // use this to add the "To" recipients.
    iMmsMtm->AddAddresseeL(_L("12345678"));

    // Setting message subject
    iMmsMtm->SetSubjectL(_L("Test MMS message"));

    // Message consists of one image
    TFileName attachmentFile;
    attachmentFile.Append(_L("c:\\data\\images\\mmsexample.jpg"));

    TMsvEntry ent = iMmsMtm->Entry().Entry();
    // Set InPreparation to false
    ent.SetInPreparation(EFalse);
    // Mark as visible, after this the message can be seen in Outbox and,
    // after sending, in Sent folder.
    ent.SetVisible(ETrue);
    iMmsMtm->Entry().ChangeL(ent);    // Commit changes

    // Save the changes
    iMmsMtm->SaveMessageL();

    // Opening store
    CMsvStore* store = iMmsMtm->Entry().EditStoreL();
    CleanupStack::PushL(store);

    // Open attachment file
    RFile attachment;
    User::LeaveIfError(attachment.Open( CEikonEnv::Static()->FsSession(),
    attachmentFile, EFileShareReadersOnly | EFileRead ));
    CleanupClosePushL( attachment );

    // Mime header
    CMsvMimeHeaders* mimeHeaders = CMsvMimeHeaders::NewL();
    CleanupStack::PushL( mimeHeaders );
    mimeHeaders->SetSuggestedFilenameL(_L("mmsexample.jpg"));

    // Represents a single attachment and information about the attachment
    CMsvAttachment* attaInfo =
        CMsvAttachment::NewL( CMsvAttachment::EMsvFile );
    CleanupStack::PushL( attaInfo );

    // Mime Type
    _LIT8(KMimeType, "image/jpeg");
    TBufC8<10> mimeType(KMimeType);

    TMsvAttachmentId attachId = KMsvNullIndexEntryId;

    // Attachment file must be an public folder (e.g. c:\Data\images)
    iMmsMtm->CreateAttachment2L(
                            *store,
                            attachment,
                            mimeType,
                            *mimeHeaders,
                            attaInfo,
                            attachId );

    CleanupStack::Pop( attaInfo ); // attaInfo
    CleanupStack::PopAndDestroy(mimeHeaders); // mimeHeaders

    store->CommitL();
    attachment.Close();
    CleanupStack::PopAndDestroy(); // attachment
    CleanupStack::PopAndDestroy(); // store


    // Start sending the message via the Server MTM to the MMS server
    CMsvOperationWait* wait = CMsvOperationWait::NewLC();
    wait->iStatus = KRequestPending;
    CMsvOperation* op = NULL;
    op = iMmsMtm->SendL(wait->iStatus );
    wait->Start();
    CleanupStack::PushL( op );
    CActiveScheduler::Start();
```


(C) Copyright Nokia 2013. All rights reserved.

```
    // The following is to ignore the completion of other active objects.
    // It is not needed if the app has a command absorbing control (using CCommandAbsorbingControl).
    while( wait->iStatus == KRequestPending )
        CActiveScheduler::Start();

    CleanupStack::PopAndDestroy(2); // op, wait
    }
```

How to use

```
iMMSSender = new (ELeave) CMMSSender();
CleanupStack::PushL(iMMSSender);
iMMSSender->ConstructL();
CleanupStack::Pop();// iMMSSender

iMMSSender->SendMMSL();
```

Postconditions

An MMS message is sent.

See also

- Size limits of MMS messages

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.