

ID	TSS001385	Creation date	May 20, 2009
Platform	S60 3rd Edition, FP1 S60 3rd Edition, FP2 S60 5th Edition	Devices	Tested on Nokia N95, Nokia E66, Nokia N82, Nokia N96, Nokia N78
Category	Symbian C++	Subcategory	Messaging, Email

Keywords (APIs, classes, methods, functions): CEmailAccounts, CImImap4Settings, CImSmtpSettings

Description

An email IMAP4/POP3 account can be created using `CEmailAccounts`. This article explains what is required to make the created email account appear in the Messaging application.

Solution

The following sample code shows how to define IMAP4 and SMTP settings, create a new email account, and display the new mailbox in the Messaging application.

```
#include <commdb.h>           // link against commdb.lib
#include <cemailaccounts.h> // link against imcm.lib
#include <cdbpreftable.h>    // for CCommsDbConnectionPrefTableView
#include <iapprefs.h>       // for CImIAPPreferences, TImIAPChoice
#include <imapset.h>       // for CImImap4Settings
#include <smtpset.h>       // for CImSmtpSettings

CEmailAccounts* emailAccs = CEmailAccounts::NewLC();

// Read settings from the connection preference table
CCommsDatabase *commsDB = CCommsDatabase::NewL( EDatabaseTypeIAP );
CleanupStack::PushL( commsDB );
CCommsDbConnectionPrefTableView* apView =
    commsDB->OpenConnectionPrefTableInRankOrderLC( ECommDbConnectionDirectionUnknown );
apView->GotoFirstRecord();
CCommsDbConnectionPrefTableView::TCommDbIapConnectionPref firstPref;
apView->ReadConnectionPreferenceL( firstPref );
CleanupStack::PopAndDestroy( 2, commsDB ); // apView, commsDB

// Use the same preferences for the mail service
TImIAPChoice apChoice;
```

TSS001385_-_Creating_an_email_account_and_showing_it_in_the_Messaging_application

```
apChoice.iIAP = firstPref.iBearer.iIapId;
apChoice.iDialogPref = ECommDbDialogPrefPrompt;

// Add an access point for the mail service
CImIAPPreferences* apPrefs = CImIAPPreferences::NewLC();
apPrefs->AddIAPL( apChoice );

// Create Imap4 settings
CImImap4Settings *imap4Settings = new ( ELeave ) CImImap4Settings;
CleanupStack::PushL( imap4Settings );

_LIT8( KImap4User, "Username" );
_LIT8( KImap4Pwd, "Password" );
imap4Settings->Reset();
imap4Settings->SetLoginNameL( KImap4User );
imap4Settings->SetPasswordL( KImap4Pwd );
imap4Settings->SetAutoSendOnConnect( EFalse );
imap4Settings->SetImapIdle( ETrue );
imap4Settings->SetDisconnectedUserMode( ETrue );
imap4Settings->SetDeleteEmailsWhenDisconnecting( EFalse );
imap4Settings->SetAcknowledgeReceipts( EFalse );
imap4Settings->SetMaxEmailSize( KMaxTInt );
imap4Settings->SetGetMailOptions( EGetImap4EmailHeaders );

// Incoming mail server settings
_LIT( KImap4SrvAddress, "mail.address" );
_LIT( KAccountName, "Imap4 account" );
imap4Settings->SetServerAddressL( KImap4SrvAddress );
imap4Settings->SetPort( KIMAPDefaultPortNumber );
imap4Settings->SetSecureSockets( EFalse );

TImapAccount imapaccount =
    emailAccs->CreateImapAccountL( KAccountName,
                                   *imap4Settings,
                                   *apPrefs, EFalse );

// Create SmtP settings
CImSmtPSettings *smtPSettings = new ( ELeave ) CImSmtPSettings;
CleanupStack::PushL( smtPSettings );
smtPSettings->Reset();
_LIT( KEmailAddress, "myaddr@smtP.address");
_LIT( KReplyToAddress, "myaddr@smtP.address");
_LIT( KReceiptAddress, "myaddr@smtP.address");
_LIT8( KSmtPLogin, "loginname" );
_LIT8( KSmtPPass, "password");
smtPSettings->SetEmailAddressL( KEmailAddress );
smtPSettings->SetReplyToAddressL( KReplyToAddress );
smtPSettings->SetReceiptAddressL( KReceiptAddress );
smtPSettings->SetLoginNameL( KSmtPLogin );
smtPSettings->SetPasswordL( KSmtPPass );
smtPSettings->SetRequestReceipts( ETrue );
smtPSettings->SetSendCopyToSelf( ESendCopyAsBccRecipient );
smtPSettings->SetSendMessageOption( ESendMessageImmediately );

// Outbound mail server settings
_LIT( KServerAddress, "serversmtP.address");
smtPSettings->SetServerAddressL( KServerAddress );
smtPSettings->SetPort( KSMTMPDefaultPortNumber );
smtPSettings->SetSecureSockets( EFalse );

// Create SmtP account
TSmtPAccount smtPaccount =
```

TSS001385_-_Creating_an_email_account_and_showing_it_in_the_Messaging_application

```
emailAccs->CreateSmtplibAccountL( imapaccount,
                                   *smtpSettings,
                                   *apPrefs,
                                   EFalse );

emailAccs->SetDefaultSmtplibAccountL( smtpaccount );

CleanupStack::PopAndDestroy( 4 ); // smtpSettings, imap4Settings,
                                   // apPrefs, emailAccs
```

Note that if some SMTP settings are not set, the Messaging application crashes when the created mailbox is opened manually in some S60 3rd Edition, FP1 devices.

The following piece of code allows the new mailbox to be shown in the Messaging application. The class from which this code is called should implement the `MMsvSessionObserver` interface.

```
#include <msvapi.h>           // link against msgs.lib

CMsvSession* msgSession = CMsvSession::OpenSyncL( *this );
CleanupStack::PushL( msgSession );
_LIT( KDescription, "dummy" );
CMsvEntry* cEntry = msgSession->GetEntryL( imapaccount.iImapService );
CleanupStack::PushL( cEntry );
TMsvEntry tEntry = cEntry->Entry();
tEntry.iDescription.Set( KDescription );
cEntry->ChangeL( tEntry );
CleanupStack::PopAndDestroy( cEntry );
```

Required capabilities: ReadDeviceData, WriteDeviceData