



<b>ID</b>	CS001056	<b>Creation date</b>	July 1, 2008
<b>Platform</b>	S60 3rd Edition, FP1	<b>Tested on devices</b>	Nokia N93
<b>Category</b>	Symbian C++	<b>Subcategory</b>	Files/Data

**Keywords (APIs, classes, methods, functions):** CAknMultiLineDataQueryDialog, CAknTextQueryDialog, NUMSECRETED, SECRETED, CAknMultiLineDataQueryDialog::ExecuteLD, CAknTextQueryDialog::ExecuteLD

## Overview

This code snippet shows how to create a password dialog with secret editors. Numeric secret editors can be defined using s NUMSECRETED resource structure in a resource file and alphanumeric secret editors are defined using a SECRETED resource structure. This code snippet implements two example dialogs with SECRETED structures. The first one is a multiline query dialog containing password and confirmation fields and the second one is a simple text query dialog with one password field.

This snippet can be self-signed.

## MMP file

The following libraries are required:

```
LIBRARY    avkon.lib    //Avkon resources
```

## Resource file

```
.RSS

#define KMaxPasswordLength 10

RESOURCE DIALOG r_dialog_password_and_confirmation_query
{
    flags = EGeneralQueryFlags;
```

## CS001056\_-\_Creating\_a\_password\_dialog\_with\_secret\_editors

```
buttons = R_AVKON_SOFTKEYS_OK_CANCEL;
items =
{
    DLG_LINE
    {
        type = EAknCtMultilineQuery;
        id = EMultilineFirstLine;
        control = AVKON_DATA_QUERY
        {
            layout = EMultiDataFirstSecEd;
            label = "Enter Password:";
            control = SECRETED
            {
                num_letters = KMaxPasswordLength;
            };
        };
    },
    DLG_LINE
    {
        type = EAknCtMultilineQuery;
        id = EMultilineSecondLine;
        control = AVKON_DATA_QUERY
        {
            layout = EMultiDataSecondSecEd;
            label = "Confirm Password:";
            control = SECRETED
            {
                num_letters = KMaxPasswordLength;
            };
        };
    }
};
};
}
```

```
RESOURCE DIALOG r_dialog_password_query
{
    flags = EAknGeneralQueryFlags;

    buttons = R_AVKON_SOFTKEYS_OK_CANCEL;
    items =
    {
        DLG_LINE
        {
            type = EAknCtQuery;
            id = EGeneralQuery;
            control = AVKON_DATA_QUERY
            {
                layout = ECodeLayout;
                label = "Enter Password:";
                control = SECRETED
                {
                    num_letters = KMaxPasswordLength;
                };
            };
        }
    };
};
}
```

## Header file

```
#ifndef __PASSWORDAPPUI_H__
#define __PASSWORDAPPUI_H__

const TInt KMaxPasswordLength = 10;
//...

class CPasswordAppUi : public CAknAppUi
{
//...
private:
//...
    TBool ShowPasswordAndConfirmationDialogL(TDes& aPassword);
    TBool ShowPasswordDialogL(TDes& aPassword);
    void UsePasswordDialogsL(); //a test method for dialogs
private:
//...
    HBufC* iPassword;
};

#endif // __PASSWORDAPPUI_H__
```

## Source file

```
#include <AknQueryDialog.h>
#include <Password_0xE0859401.rsg>

//...
CPasswordAppUi::CPasswordAppUi() : iPassword(NULL)
{
}

CPasswordAppUi::~CPasswordAppUi()
{
//...
    if (iPassword)
    {
        delete iPassword;
        iPassword=NULL;
    }
}

TBool CPasswordAppUi::ShowPasswordAndConfirmationDialogL(TDes& aPassword)
{
    TBuf<KMaxPasswordLength> password;
    TBuf<KMaxPasswordLength> confirmation;

    CAknMultiLineDataQueryDialog* dlg =
    CAknMultiLineDataQueryDialog::NewL(password, confirmation);

    if (!dlg->ExecuteLD(R_DIALOG_PASSWORD_AND_CONFIRMATION_QUERY))
    {
        //Cancel key pressed
        //do something...
        return EFalse;
    }
}
```

## CS001056\_-\_Creating\_a\_password\_dialog\_with\_secret\_editors

```
    }
else
{
    if(password.Compare(confirmation) == 0)
    {
        //password and confirmedPassword match.
        //do something...

        aPassword.Copy(password);
        return ETrue;
    }
    else if(password.Length() != confirmation.Length())
    {
        //password buffer and confirmation buffer have different lengths
        //do something...

        return EFalse;
    }
    else
    {
        //password buffer and confirmation buffer have different contents
        //do something...

        return EFalse;
    }
}

TBool CPasswordAppUi::ShowPasswordDialogL(TDes& aPassword)
{
    TBuf<KMaxPasswordLength> password;

    CAknTextQueryDialog* dlg =
    new(ELeave) CAknTextQueryDialog(password, CAknQueryDialog::ENoTone );

    if(!dlg->ExecuteLD(R_DIALOG_PASSWORD_QUERY))
    {
        //Cancel key pressed
        //do something...
        return EFalse;
    }
    else
    {
        //OK do something...
        aPassword.Copy(password);
        return ETrue;
    }
}

void CPasswordAppUi::UsePasswordDialogsL()
{
    //reset old password if exists
    if(iPassword)
    {
        delete iPassword;
        iPassword=NULL;
    }

    iPassword = HBufC::NewL(KMaxPasswordLength);
    TPtr bufPtr = iPassword->Des();
}
```

## CS001056\_-\_Creating\_a\_password\_dialog\_with\_secret\_editors

```
//show password and confirmation dialog and store the given
//password in member variable iPassword if input values are ok
TBool ret = ShowPasswordAndConfirmationDialogL(bufPtr);

if(ret)
{
    //create a temporary buffer to store the given password
    HBufC* tmpPassword = HBufC::NewLC(KMaxPasswordLength);
    TPtr tmpPtr = tmpPassword->Des();

    //show a password dialog
    TBool ret2 = ShowPasswordDialogL(tmpPtr);

    if(ret2)
    {
        //compare the given password with member variable iPassword
        if (tmpPassword->Compare(*iPassword) == 0)
        {
            //passwords match ok, do something....
        }
        else
        {
            //The given password is not correct, do something...
        }
    }
    else
    {
        //the user pressed cancel, there is no password to compare
        //do something...
    }

    CleanupStack::PopAndDestroy(); //tmpPassword
}
else
{
    //giving a new password failed, do something...
    delete iPassword;
    iPassword=NULL;
}
}
```

## Postconditions

The example code shows a password and confirmation dialog to the user and stores the given password into the member variable iPassword. After this, a simple password query dialog is shown to the user.