



Contents

- [1 What is Form](#)
- [2 lib in mmp file](#)
- [3 header files included](#)
- [4 Creating Form from Resource](#)
- [5 Header File](#)
- [6 Implementation File](#)
 - ◆ [6.1 Getting Form Control Value](#)
 - ◆ [6.2 Setting Form Control Value](#)
- [7 Executing Form](#)
- [8 Getting Key Press Event in Form and Exiting from Form](#)
- [9 Adding Menus to Form](#)
- [10 Hiding the Default Menus](#)
- [11 Processing commands](#)
- [12 Adding Control Dynamically](#)
- [13 Setting Focus on Desired Control on Form](#)
- [14 Updating Form](#)
- [15 Getting Control ID in Focus](#)
- [16 Getting the Mode \(editable or view mode \)](#)
- [17 Common Mistakes](#)
 - ◆ [17.1 Forgetting to call Base Class ConstructL\(\)](#)
 - ◆ [17.2 Empty implementation of PrepareLC \(TInt aResourceId\)](#)

What is Form

Forms provide a way to allow the user to quickly and easily enter or edit many items of data in one process. Forms can have a number of fields, displayed in a similar way to a list, which the user can scroll up and down. Unlike list items, form lines are editable when they are in focus.

Forms have either a view mode and an edit mode, or just an edit mode. Use of a view mode implies that there is some existing information to display in the form, which the user can then edit. For example, in a game application a form could be used in view mode to display information about existing details for an opponent, and then in edit mode to change the details.

lib in mmp file

```
LIBRARY eikdlg.lib
```

header files included

```
#include <aknform.h>
#include <eikedwin.h>
#include <e32cmn.h>
```

Creating Form from Resource

Forms are contained within a DIALOG resource, specifying the form components.

```
RESOURCE DIALOG r_myform_form_dialog
{
    = EEikDialogFlagNoDrag |
      EEikDialogFlagFillAppClientRect
      | EEikDialogFlagWait
      | EEikDialogFlagCbaButtons

    = RTAWKON_SOFTKEYS_OPTIONS_BACK;
    = r_myform_form;
}

// r_myform_form is defined as :

RESOURCE FORM r_myform_form
{
    = fEEikFormUseDoubleSpacedFormat ;
    = items
{
    DLG_LINE
{
    =EEikCtEdwin;          type
    =NAME_TEXT;           prompt
    =MyformDlgCidEdwin;   id
    = EEikDlgItemTakesFlagsKey | EEikDlgItemOfferAllHotKeys;
    =EDWIN                control
{
    =50;                  width
    =50;                  maxlength
};
},
    DLG_LINE
{
    =EEikCtEdwin;          type
    =PHONE_NO_TEXT;       prompt
    =MyformDlgCidEdwin2;  id
    = EEikDlgItemTakesFlagsKey | EEikDlgItemOfferAllHotKeys;
    =EDWIN                control
{
    =50;                  width
    =50;                  maxlength
};
},
    DLG_LINE
```

Forms_in_Symbian_c++

```
{
    =EEikCtEdwin;          type
    =EMAIL_TEXT;          prompt
    =EMyformDlgCIdEdwin3; id
    =EDWIN                 control
{
    =50;                   width
    =50;                   maxlength
};
}
};
}
```

Header File

```
// Derive the class from CAknForm

class CMyForm : public CAknForm
{
public: // Constructor and destructor
    /**
     * NewL
     * Two-phased constructor.
     */
    static CMyForm* NewL();

    /**
     * ~CMyForm
     * Destructor.
     */
    virtual ~CMyForm();

public:
    /**
     * From CAknForm, ExecuteLD
     * @return CAknForm::ExecuteLD return value
     * @param aResourceId resource ID
     */
    TInt ExecuteLD( TInt aResourceId );

    /**
     * From CAknForm, PrepareLC
     * @param aResourceId resource ID
     */
    void PrepareLC( TInt aResourceId );

private: // Constructor
    /**
     * CMyForm
     * Default constructor.
     */
    CMyForm();

    /**
     * ConstructL
     * Second-phase constructor.
     */
}
```

Forms_in_Symbian_c++

```
void ConstructL();

private: // Functions from base class
/**
 * From CEikDialog, PostLayoutDynInitL
 * Set default field value to member data.
 */
void PreLayoutDynInitL();

/**
 * From CAknForm , QuerySaveChangesL
 * Show save query. If user answers "No" to this query.
 * return field value to the value which is before editing.
 */
TBool QuerySaveChangesL();

/**
 * From CAknForm , SaveFormDataL
 * Save the contents of the form.
 */
TBool SaveFormDataL();

/**
 * From CAknForm, DoNotSaveFormDataL
 * Does not save the contents of the form.
 */
void DoNotSaveFormDataL();

private:
/**
 * For holding controls value
 */
TBuf<EAknExFormEdwinMaxLength> iDataName;
.
.
.
.
};
```

Implementation File

```
/**
 * Symbian OS 2 phase constructor.
 * Constructs the CMyForm using the NewLC method, popping
 * the constructed object from the CleanupStack before returning it.
 *
 * @return The newly constructed CMyForm
 */
CMyForm * CMyForm ::NewL()
{
    CMyForm* cmyform= new (ELeave) CMyForm ();
    CleanupStack::Push(self);
    ->ConstructL();
    CleanupStack::Pop(self);
    return self;
}
```

Getting Form Control Value

```

/**
 * Called by the framework whenever the 'Save' menu item is selected, and by the
 * QuerySaveChangesL method when the user answers yes to the save query.
 * Saves the data from the forms controls.
 * @return TBool ETrue if the form data has been saved, EFalse otherwise.
 */

TBool CMyForm ::SaveFormDataL()
{
    CEikEditor = static_cast <CEikEditor*> (ControlOrNull(EmyformDlgCidEdwin));
    if ( nEditor )
    {
        // Save the value of Control to class member //
    }

    return ETrue;
}

```

Setting Form Control Value

```

/**
 * Called when the form is executed and when the user chooses to discard changes
 * in QuerySaveChangesL (via DoNotSaveFormDataL).
 */

void CMyForm ::LoadFormValuesFromDataL()
{
    CEikEditor = static_cast <CEikEditor*> (ControlOrNull(EmyformDlgCidEdwin));
    if (nEditor)
    {
        // Example : set the Control Text //

        <50> myName;          TBuf
        Copy(_L("Vasant"), myName.
        * name = myName.AllABC();
        // Setting the Value //

        ->SetTextL(name);nEditor
        //CleanupStack::PopAndDestroy(name);

    }

}

/**
 * Called by QuerySaveChangeL when the user chooses to discard changes made to
 * the form. Loads the form values from iOpponent
 */

void CMyForm ::DoNotSaveFormDataL()
{
    LoadFormValuesFromDataL
}

```

```

/**

```

Forms_in_Symbian_c++

```
* Called by the framework before the form is initialised
* Loads the form values from iOpponent ready for execution of the form
*/

void CMyForm ::PreLayoutDynInitL()
{
    CAknForm::PreLayoutDynInitL();
    LoadFormValuesFromDataL();
}

/**
* Called by the framework when a menu is displayed.
* Removes the default items from the options menu of the form for editing a
* fields label, adding a field and deleting a field
*/

void CMyForm ::DynInitMenuPaneL(TInt aResourceId, CEikMenuPane* aMenuPane)
{
    CAknForm::DynInitMenuPaneL(aResourceId, aMenuPane);

    if (aResourceId == R_AVKON_FORM_MENUPANE)
    {
        ->SetItemEnabled(EAknFormCmdLabel, ETrue);
        ->SetItemEnabled(EAknFormCmdAdd, ETrue);
        ->SetItemEnabled(EAknFormCmdDelete, ETrue);
    }
}
```

Executing Form

```
CMyForm* form = CMyForm ::NewL();
TInt ret = form->ExecuteLD( R_MYFORM_FORM_DIALOG );
```

Getting Key Press Event in Form and Exiting from Form

```
// override OfferKeyEventL for getting key press events. //

TKeyResponse CMyForm::OfferKeyEventL(const TKeyEvent& aKeyEvent, TEventCode aType)
{
    // We send EKeyEscape, for exiting from form if user press EKeyLeftArrow

    if ( aKeyEvent.iCode == EKeyLeftArrow )
    {
        ; TKeyEvent key
        iCode = EKeyEscape; key.
        iScanCode = EStdKeyEscape;
        iRepeats = 0; key.
        iModifiers = 0; key.
        return CAknForm::OfferKeyEventL(key, EEventKey);
    }
}
```

Adding Menus to Form

```
// Override DynInitMenuPaneL //

void CMyForm::DynInitMenuPaneL(TInt aResourceId, CEikMenuPane* aMenuPane)
{
    CAknForm::DynInitMenuPaneL(aResourceId, aMenuPane);

    if (aResourceId == R_AVKON_FORM_MENUPANE)
    {
        // Add the menu from the resource defined in the .rss file
        ->AddMenuItemsL(MENUBARFORM_ASS_MENU, 0, 1);
    }
}
```

Hiding the Default Menus

```
void CMyForm::DynInitMenuPaneL(TInt aResourceId, CEikMenuPane* aMenuPane)
{
    CAknForm::DynInitMenuPaneL(aResourceId, aMenuPane);

    if (aResourceId == R_AVKON_FORM_MENUPANE)
    {
        // you can hide the default form menus //

        ->SetItemDimmed(MENUBARFORM_CmdLabel, ETrue);
        ->SetItemDimmed(MENUBARFORM_CmdAdd, ETrue);
        ->SetItemDimmed(MENUBARFORM_CmdDelete, ETrue);
    }
}
```

Processing commands

```
/*
 * ProcessCommandL() can be overridden to handle * custom commands. Built-in form
 * commands are * handled by CAknForm.
 */

void CMyForm::ProcessCommandL( TInt aCommandId )
{
    // Form default commands.
    CAknForm::ProcessCommandL( aCommandId );
    // Custom commands.
    switch ( aCommandId )
    {
        case ECommand1:
            User::InfoPrint( _L("Pressed command1 on Form") );
            break;
        default:
            break;
    }
}
```

```

    }
}

```

Adding Control Dynamically

In edit mode, fields can be added or deleted dynamically using `CAknForm::AddItemL()`

```

void CMyForm::AddItemL()
{
    _LIT( caption, "New Control" );
    TInt pageId = ActivePageId();
    TInt id = iTotalControls++;
    TInt type = EEikCtEdwin;
    TAny* unused=0;
    CEikEdwin* edwin = (CEikEdwin*)CreateLineByTypeL
        ( caption, pageId, id, type, unused );
    edwin->ConstructL
        ( EEikEdwinNoHorizScrolling | EEikEdwinResizable, 10, 100, 1 );
    Line( id )->ActivateL();
    SetEditableL( IsEditable() );
    DrawNow();
}

```

Setting Focus on Desired Control on Form

```

// Pass the Id of the Control on which focus is to be set.
TryChangeFocusToL( EmyformDlgCIdEdwin2 );

```

Updating Form

```

// Only has effect on forms, call this after adding or deleting lines
UpdatePageL( ETrue );

```

Getting Control ID in Focus

```

// Gets the ID of the Control having Focus //
TInt FocusedControlId = IdOfFocusControl();
if( FocusedControlId == EmyformDlgCIdEdwin)
{
    // Do Something //
}

```

Getting the Mode (editable or view mode)

```

if( IsEditable() )
{
    // Form is in the edit mode //
}
else
{
    // Form is not in the edit mode //
}

```

Common Mistakes

When deriving from CAknForm, people generally makes some mistakes, which are hard to spot and which generally does not give any error and hence are hard to spot:

Forgetting to call Base Class ConstructL()

Remember that 'C' class implements two phase construction and hence a class derived from 'C' class needs to call second phase construction of the base class:

```

void CYourForm::ConstructL()
{
    CAknForm::ConstructL();
}

```

Empty implementation of PrepareLC (TInt aResourceId)

If you are not interested in your own implementation then one should not have this function, let the default implementation of base class handle it, as it is called after ExecuteLD(). So either remove it from your code or at least pass the call to the base class in it, something like below:

```

void CYourForm::PrepareLC (TInt aResourceId)
{
    CAknForm::PrepareLC (aResourceId);
}

```