



Following code sample shows how you could use progress and wait notes with your applications. Note that only the effected functions are shown here, thus you are expected to copy-paste the functions to your own class definition.

With this example StartWaitNoteL() can be used to start showing standard wait dialog, when the progress length can not be estimated, for the progress dialog you need to supply the final value as an argument variable when using StartProgressNoteL()-function.

To show the progress with progress dialog you could use UpdateProcessL() to supply the current progress value along side with the progress text.

The callback interface function DialogDismissedL() is called not only if user presses cancel button, but also after ProcessFinishedL() is called. Therefore you should cancel any process inside DialogDismissedL() if, and only if, aButtonId equals to the key used to close dialog, in this case EAknSoftkeyCancel.

Source code

```

CMyClass::~CMyClass()
{
    if(iProgressDialog)
    {
        iProgressDialog->ProcessFinishedL();
    }

    if(iWaitDialog)
    {
        iWaitDialog->ProcessFinishedL();
    }
}

void CMyClass::StartWaitNoteL()
{
    iWaitDialog = new (ELeave) CAknWaitDialog(
        (REINTERPRET_CAST(CEikDialog**, &iWaitDialog)),
        ETrue);
    iWaitDialog->PrepareLC(R_WAIT_NOTE_SOFTKEY_CANCEL);
    iWaitDialog->SetCallback(this);
    iWaitDialog->RunLD();
}

void CMyClass::StartProgressNoteL(TInt aFinalValue)
{
    iProgressDialog = new (ELeave) CAknProgressDialog(
        (REINTERPRET_CAST(CEikDialog**, &iProgressDialog)),
        ETrue);
    iProgressDialog->PrepareLC(R_PROGRESS_NOTE);
    iProgressInfo = iProgressDialog->GetProgressInfoL();
    iProgressDialog->SetCallback(this);
    iProgressDialog->RunLD();
    iProgressInfo->SetFinalValue(aFinalValue);
}

void CMyClass::UpdateProcessL(TInt aProgress,const TDesC& aProgressText)

```

Progress_and_wait_notes

```
{
    if(iProgressDialog)
    {
        iProgressDialog->SetTextL(aProgressText);
    }

    if(iProgressInfo)
    {
        iProgressInfo->SetAndDraw(aProgress);
    }
}

void CMyClass::DialogDismissedL(TInt aButtonId)
{
    iProgressDialog = NULL;
    iProgressInfo = NULL;
    iWaitDialog = NULL;

    if (aButtonId == EAknSoftkeyCancel)
    {
        // cancel any process in here
    }
}
```

Header definitions

```
#include <eikprogi.h>
#include <aknwaitdialog.h>
#include <AknProgressDialog.h>
// Link against eikctl.lib

class CMyClass : public CBase, public MProgressDialogCallback
{
public:
    ~CMyClass();

    void StartWaitNoteL();
    void StartProgressNoteL(TInt aFinalValue);
    void UpdateProcessL(TInt aProgress,const TDesC& aProgressText);

protected:// other system interface functions
    void DialogDismissedL (TInt aButtonId);

private:
    CAknProgressDialog* iProgressDialog;
    CEikProgressInfo* iProgressInfo;
    CAknWaitDialog* iWaitDialog;
};
```

Resource definitions

```
RESOURCE_DIALOG r_wait_note_softkey_cancel
{
    flags = EAknWaitNoteFlags | EEikDialogFlagNotifyEsc;
    buttons = R_AVKON_SOFTKEYS_CANCEL;
```

Progress_and_wait_notes

```
items =
{
    DLG_LINE
    {
        type = EAknCtNote;
        id = EYBWaitNoteId;
        control= AVKON_NOTE
        {
            layout = EWaitLayout;
            animation = R_QGN_GRAF_WAIT_BAR_ANIM;
        };
    }
};

RESOURCE DIALOG r_progress_note
{
    flags = EAknProgressNoteFlags;
    buttons = R_AVKON_SOFTKEYS_CANCEL;
    items =
    {
        DLG_LINE
        {
            type = EAknCtNote;
            id = EProgressNoteId;
            control = AVKON_NOTE
            {
                layout = EProgressLayout;
                singular_label = "Progressing";
            };
        }
    };
}
```

You are also expected to have the definitions of EYBWaitNoteId and EProgressNoteId in a hrh file included to your rss file.