



This example will allow you to display a wait note till you get a http response.

In the MMP file

```
LIBRARY      avkon.lib
LIBRARY      eikcdlg.lib
LIBRARY      eikctl.lib
```

In the RSS file

```
//-----
//
//  r_wait_note_http_request
//  Display wait note
//
//-----
//
RESOURCE DIALOG r_wait_note_http_request
{
    flags=EAKnWaitNoteFlags;
    items =
        {
            DLG_LINE
            {
                type = EAKnCtNote;
                id = EWaitNoteId;
                control = AVKON_NOTE
                {
                    layout = EWaitLayout;
                    singular_label = "Please wait..."
                    imagefile = "z:\\system\\data\\avkon.mbm";
                    imageid = EMbmAvkonQgn_note_progress;
                    imagemask = EMbmAvkonQgn_note_progress_mask;
                    animation = R_QGN_GRAF_WAIT_BAR_ANIM;
                };
            }
        };
}
```

In the HEADER File

```
#ifndef __TestClass_H__
#define __TestClass_H__

#include "HTTPClientEngine.h"
#include <AknWaitNoteWrapper.h>

class TestClass: public MHTTPObserver,public MAknBackgroundProcess
{
public:

    static TestClass* NewLC();
```

Display_wait_note_between_HTTP_response_delay

```
static TestClass* NewL();

void ConstructL();

TestClass();

virtual ~TestClass();

void IssueHTTPRequest();

/*
 * HTTP Callback Methods
 *
 */

virtual void ClientEvent(const TDesC& aEventDescription);

virtual void ClientBodyReceived(const TDesC8& aBodyData);

/*
 *      From MAKnBackgroundObserver
 *
 */

TBuf<32> Status;

RTimer iTimeWaster;

private:

    TBool iDialogDismissed;

    void DialogDismissedL(TInt aButtonId);

    TBool IsProcessDone() const;

    void ProcessFinished();

    void StepL();

private:

    HBufC8*                               iHTTPResponse;

    CHTTPClientEngine*                   iHTTPClient;

    CAknWaitNoteWrapper*                 waitNoteWrapper;

};

#endif __TestClass_H__
```

In the CPP File

```
// TestClass.cpp: implementation of the CTestClassclass.
```

Display_wait_note_between_HTTP_response_delay

```
//
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
#include "TestClass.h"
#include <e32std.h>
#include <eikmenup.h>
#include <e32base.h>
#include <aknotewrappers.h>
#include <StringLoader.h>
/*
=====
Name      : CTestClass from TestClass.h
Author    : shivam
Version   :
=====
*/

CTestClass ::CTestClass()
{

}

CTestClass::~CTestClass()
{

    if(iHTTPClient)
    {
        delete iHTTPClient;
        iHTTPClient=NULL;
    }

    if(iHTTPResponse)
    {
        delete iHTTPResponse;
        iHTTPResponse=NULL;
    }

    iTimeWaster.Close();

    delete this->waitNoteWrapper;
    this->waitNoteWrapper=NULL;

}

void CTestClass::CTestClass()
{
    iHTTPClient=CHTTPClientEngine::NewL(*this);

    iHTTPResponse=NULL;

    waitNoteWrapper = CAknWaitNoteWrapper::NewL();
    iTimeWaster.CreateLocal();

}

CTestClass* CTestClass::NewL()
```

Display_wait_note_between_HTTP_response_delay

```
{
    CTestClass* self = CTestClass::NewLC();
    CleanupStack::Pop(self);
    return self;
}

CTestClass* CTestClass::NewLC()
{
    CTestClass* self = new (ELeave) CTestClass();
    CleanupStack::PushL(self);
    self->ConstructL();
    return self;
}

//=====
//ClientBodyReceived() -- HTTP Callback Function.This will get the response
// body data.
//returns -- None
//
//=====

void CTestClass::ClientBodyReceived(const TDesC8& aBodyData)
{
    if(!iHTTPResponse)
        {
            iHTTPResponse = HBufC8::NewL(aBodyData.Length());
        }
    else
        {
iHTTPResponse =
    iHTTPResponse->ReAllocL(iHTTPResponse->Length()+aBodyData.Length());
        }

    TPtr8 ptr_bufNoteData8=iHTTPResponse->Des();
    ptr_bufNoteData8.Append(aBodyData);
}

//=====
//ClientEvent() -- HTTP Callback function
//aEventDescription -- The transaction events
//returns -- None
//
//=====

void CTestClass::ClientEvent(const TDesC& aEventDescription)
{
    this->Status.Copy(aEventDescription);

    _LIT(KTransactionSuccessful, "Transaction Successful");
    if(aEventDescription==KTransactionSuccessful)
        {
            //Handle HTTP Response
        }
}

//=====
```

Display_wait_note_between_HTTP_response_delay

```
//
//IssueHTTPRequest() -- Issue HTTP request
//
//
//=====

void CConnector::IssueHTTPRequest()
{
    iRSSFeed = aCurrentFeed;

    Status.Copy(_L("Init"));

    TBuf8<256> uri8;
    uri8.Append(_L("http://www.forum.nokia.com"));

    // Start transaction
    iHTTPClient->IssueHTTPGetL(uri8);

if (!waitNoteWrapper->ExecuteL(R_WAIT_NOTE_HTTP_REQUEST, *this, ETrue))
{
    iHTTPClient->CancelTransaction();
}

}
//=====
//StepL() -- Used by the Wait bar to increment the timer
//Inherited from MAknBackgroundObserver
//
//=====

void CTestClass::StepL()
{
    TRequestStatus status;
    TInt delay = 100000; // 1 second
    iTimeWaster.After(status, delay);
    User::WaitForRequest(status);
}

//=====
//ProcessFinished() -- Diminishes the wait bar when the http process is finished
//
//Inherited from MAknBackgroundObserver
//=====

void CTestClass::ProcessFinished()
{
    iTimeWaster.Cancel();
    this->Status.Delete(0, this->Status.Length());
}

//=====
//IsProcessDone() -- Checks whethet the HTTP request is complete or not
//returns : True/False
//Inherited from MAknBackgroundObserver
//=====

TBool CTestClass::IsProcessDone() const
```

Display_wait_note_between_HTTP_response_delay

```
{
    _LIT(KTransactionSuccessful, "Transaction Successful");
    TPtrC istatus(this->Status);
    return(istatus==KTransactionSuccessful);
}

//=====
//DialogDismissedL() -- Called if the wait bar is stoped in between
//Inherited from MAknBackgroundObserver
//
//=====

void CTestClass::DialogDismissedL(TInt /*aButtonId*/)
{
}
}
```