

How_to_get_the_count_of_recent_missed_calls

ID		Creation date	March 20, 2008
Platform	S60 3rd, S60 3rd FP1	Tested on devices	N73, N95 8Gb
Category	Symbian C++	Subcategory	Calls
Keywords: Recent missed calls			

Overview

The original article is in Russian. You can find it [here](#).

You can use class **CMissedCallCalc** for getting the count of recent missed calls and for tracking appearance of new missed calls.

As all active objects, this class emulates final state machine (method `RunL()`). The set of states is defined in enum `TState`.

This class uses the *Log Engine*. Asynchronous method `NotifyChange()` of class **CLogEngine** is used for notify about new changes. When the new changes has occurred in database - `RunL()` called. Class **CLogViewRecent** is used for further asynchronous reading log.

Main methods:

- `Count()` - returns count of the new recent calls
- `IsCalcCompleted()` - returns check result (calculation is completed or not)

Header File

```
#include <e32base.h>
#include <logcli.h>
#include <logview.h>

const TInt KTimeDelay = 1000000; // minimum timeout for notification request

class CMissedCallCalc : public CActive
{
    enum TState
    {
        EWaitingChange = 1,
        EReadingLog,
        EReadingLogItems,
        EFindingDuplicates,
        EReadingDupeItems
    };

public:
    static CMissedCallCalc* NewL(); // factory
    ~CMissedCallCalc();

    inline TInt Count() { return iCount; } // missed call count
```

How_to_get_the_count_of_recent_missed_calls

```
inline TBool IsCalcCompleted() { return iState == EWaitingChange; } // if true - then Count

protected:
    CMissedCallCalc();
    void ConstructL();

    void GetLatestL();
    void StartL();
    void DoCancel();
    void RunL();

    void ReadPreviousL();

private:
    TInt iCount; // current count of the missed calls
    TInt iDupeCount; //used for interim count of duplicates
    TState iState; // current state

    CLogClient* iLogClient;
    CLogViewRecent* iRecentLogView;
    RFs iFsSession;

    //to detect duplicates
    CLogViewDuplicate* iDuplicateView;
};
```

Source File

```
#include "callCalc.h"

CMissedCallCalc* CMissedCallCalc :: NewL()
{
    CMissedCallCalc* self = new (ELeave) CMissedCallCalc();
    CleanupStack :: PushL( self );
    self->ConstructL();
    CleanupStack :: Pop(self);
    return self;
}

CMissedCallCalc :: CMissedCallCalc(): CActive( CActive :: EPriorityStandard )
{
}

CMissedCallCalc :: ~CMissedCallCalc()
{
    Cancel();

    delete iRecentLogView;
    iRecentLogView = NULL;

    delete iDuplicateView;
    iDuplicateView = NULL;

    delete iLogClient;
    iLogClient = NULL;

    iFsSession.Close();
}
```

How_to_get_the_count_of_recent_missed_calls

```
void CMissedCallCalc :: ConstructL()
{
    CActiveScheduler :: Add( this );

    User::LeaveIfError( iFsSession.Connect() );

    iLogClient = CLogClient :: NewL( iFsSession );
    iRecentLogView = CLogViewRecent :: NewL( *iLogClient );
    iDuplicateView = CLogViewDuplicate::NewL(*iLogClient);

    GetLatestL();
}

void CMissedCallCalc :: DoCancel()
{
    if( iRecentLogView )
        iRecentLogView->Cancel();

    if( iLogClient )
    {
        if( iState == EWaitingChange )
            iLogClient->NotifyChangeCancel();
        else
            iLogClient->Cancel();
    }
}

void CMissedCallCalc :: RunL()
{
    if( iStatus != KErrCancel )
        switch( iState )
        {
            case EWaitingChange: // new event
                GetLatestL();
                break;

            case EReadingLog: // start reading log events from last to first
                if( iRecentLogView->CountL() > 0 )
                {
                    iCount = 0; // clear value
                    iState = EReadingLogItems;
                    if( iRecentLogView->LastL( iStatus ) ) // to last event
                        SetActive();
                    else
                        StartL();
                }
                else
                    StartL();
                break;

            case EReadingLogItems: // reading event
                if( iStatus == KErrNone && iRecentLogView )
                {
                    TLogFlags iFlags = iRecentLogView->Event().Flags();
                    if( !( iFlags & KLogEventRead ) )
                    {
                        //we are going to check to see if we have a Duplicate
                        iState = EFindingDuplicates;
                        if (iRecentLogView->DuplicatesL(*iDuplicateView, iStatus))
                        {

```

How_to_get_the_count_of_recent_missed_calls

```
        SetActive();
    }
    else
    {
        //for some reason if you missed x calls from number y then you
        //answered one we fail the call to DuplicatesL and count is then off
        //(if there were duplicates)
        //should still add one for the one we missed
        iCount++;
        //then continue going "previous" in the list
        ReadPreviousL();
    }
}
else
{
    ReadPreviousL();
}
}
else
    StartL();
break;

case EFindingDuplicates:
    iDupeCount = 0;
    if( iStatus == KErrNone && iRecentLogView
        && iDuplicateView && iDuplicateView->CountL() > 0 )
    {
        iState = EReadingDupeItems;
        if( iDuplicateView->LastL( iStatus ) ) // to last event
        {
            SetActive();
        }
        else
        {
            //didn't manage to go through list of dupes, but we had at least
            //one missed call, so increment here
            iCount++;
            //anyway, we're done with duplicates
            ReadPreviousL();
        }
    }
    else
    {
        //there was no duplicate but there was one, right?
        iCount++;
        //anyway, we're done with duplicates
        ReadPreviousL();
    }
    break;

case EReadingDupeItems:
    if( iStatus == KErrNone && iDuplicateView )
    {
        TLogFlags iFlags = iDuplicateView->Event().Flags();
        if( !( iFlags & KLogEventRead ) )
        {
            iDupeCount++;
        }
        else
        {
            //since the dupe was read and is more recent than previous ones
```

How_to_get_the_count_of_recent_missed_calls

```
        //the older unread ones don't actually count but clearly the count
        //is never going below 1
        iDupeCount = 1;
    }
    //then continue going "previous" in the list
    if( iDuplicateView->PreviousL( iStatus ) ) // try to read prev. event
    {
        SetActive();
    }
    else
    {
        //at this point we add our number of dupes to our count
        //if for some reason dupes is zero we still add one since that
        //was the number we must have
        if (iDupeCount == 0)
        {
            iCount++;
        }
        else
        {
            iCount += iDupeCount;
        }
        //at this point we have got all we can from the dupes
        //return to the main list
        ReadPreviousL();
    }
}
else
{
    //at this point we add our number of dupes to our count
    //if for some reason dupes is zero we still add one since that
    //was the number we must have
    if (iDupeCount == 0)
    {
        iCount++;
    }
    else
    {
        iCount += iDupeCount;
    }
    //at this point we have got all we can from the dupes
    //return to the main list
    ReadPreviousL();
}
break;

default:
    StartL();
    break;
}
}

void CMissedCallCalc :: ReadPreviousL()
{
    iState = EReadingLogItems;
    if( iRecentLogView->PreviousL( iStatus ) ) // try to read prev. event
    {
        SetActive();
    }
    else
    {
        StartL();
    }
}
```

How_to_get_the_count_of_recent_missed_calls

```
    }  
}  
  
void CMissedCallCalc :: StartL()  
{  
    if( iRecentLogView )  
        iRecentLogView->Cancel();  
  
    iLogClient->Cancel();  
  
    iState = EWaitingChange;  
    iLogClient->NotifyChange( TTimeIntervalMicroSeconds32( KTimeDelay ), iStatus );  
    SetActive();  
}  
  
void CMissedCallCalc :: GetLatestL()  
{  
    iState = EReadingLog;  
    iRecentLogView->Cancel();  
    if( iRecentLogView->SetRecentListL( KLogRecentMissedCalls, iStatus ) )  
        SetActive();  
    else  
        StartL();  
}
```

Internal Links

- [Logs Example](#)
- [Logs monitoring Example](#)