Monitorando status da chamada com CTelephony

Este artigo ilustra como monitorar o status da chamada usando **CTelephony**. Este exemplo deve ser utilizado apenas em dispositivos S60 3rd Edition, e não funciona em dispositivos anteriores.

Para chamadas recebidas ou discadas você pode usar este código para recuperar o número discado.

Enum CTelephony::TCallStatus define possíveis status da chamada:

```
enum TCallStatus
{
    /**
    Indicates that the status is unknown.
    */
    EStatusUnknown,

    /**
    Idle line status (no active calls).
    */
    EStatusIdle,

    /**
    Call dialling status .
    */
    EStatusDialling,

    /**
    Call ringing status.
    */
    EStatusRinging,

    /**
    Call answering status.
    */
    EStatusAnswering,

    /**
    Call connecting status.
    */
    EStatusConnecting,

    /**
    Call connected status.
    */
    EStatusConnected,

    /**
    Call is undergoing temporary channel loss and it may or may not be reconnected.
    */
    EStatusReconnectPending,

    /**
    Call disconnecting status.
    */
    EStatusDisconnecting,

    /**
    Call on hold.
    */
    EStatusHold,

    /**
    Call is transferring.
    */
    EStatusTransferring,

    /**
    Call in transfer is alerting the remote party.
    */
    EStatusTransferAlerting
};
```

Realize a seguinte linkagem:

```
LIBRARY etel3rdparty.lib
```

CallsMonitor.cpp

```
#include "CallsMonitor.h"

CCallMonitor* CCallMonitor::NewLC(MCallCallBack& aObserver)
{
    CCallMonitor* self = new (ELeave) CCallMonitor(aObserver);
    CleanupStack::PushL(self);
    self->ConstructL();
    return self;
}

CCallMonitor* CCallMonitor::NewL(MCallCallBack& aObserver)
{
    CCallMonitor* self = CCallMonitor::NewLC(aObserver);
    CleanupStack::Pop(); // self;
    return self;
}


CCallMonitor::CCallMonitor(MCallCallBack& aCallBack)
:CActive(EPriorityStandard),iCallBack(aCallBack),iCurrentStatusPckg(iCurrentStatus)
{
```


(C) Copyright Nokia 2013. All rights reserved.

```
    CActiveScheduler::Add(this);
}

CCallMonitor::~CCallMonitor()
{
    Cancel();
    delete iTelephony;
}

void CCallMonitor::ConstructL(void)
{
    iTelephony = CTelephony::NewL();
    StartListening();
}

void CCallMonitor::CancelOperation(void)
{
    Cancel();
}

void CCallMonitor::DoCancel()
{
    iTelephony->CancelAsync(CTelephony::EVoiceLineStatusChangeCancel);
}


void CCallMonitor::RunL()
{
    iCallBack.CallStatusChangedL(iCurrentStatus.iStatus,iStatus.Int());
    if(iStatus != KErrCancel)
        StartListening();
}

void CCallMonitor::StartListening()
{
    Cancel();
    iCurrentStatus.iStatus = CTelephony::EStatusUnknown;
    iTelephony->NotifyChange(iStatus,CTelephony::EVoiceLineStatusChange,iCurrentStatusPckg);
    SetActive();
}
```

CallsMonitor.h

```
#include <Etel3rdParty.h>

class MCallCallBack
{
    public:
        virtual void CallStatusChangedL(CTelephony::TCallStatus& aStatus, TInt aError)=0;
};

class CCallMonitor : public CActive
{
    public:
        ~CCallMonitor();
        static CCallMonitor* NewLC(MCallCallBack& aObserver);
        static CCallMonitor* NewL(MCallCallBack& aObserver);

    private:
        CCallMonitor(MCallCallBack& aCallBack);
        void ConstructL();

    protected:
        void DoCancel();
        void RunL();

    private:
        void CancelOperation(void);
        void StartListening();

    private:
        MCallCallBack&                iCallBack;
        TInt                          iState;
        CTelephony::TCallStatusV1     iCurrentStatus;
        CTelephony::TCallStatusV1Pckg iCurrentStatusPckg;
        CTelephony*                   iTelephony;
};
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.