Archived:How to Answer call using CTelephony 3rd Edition

Archived: This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

Reviewer Approved

Overview

How to answer call using CTelephony in 3rd Edition.

This is a very simply task. All you have to do is to implement an Active class which watches the events of your CTelephony object.

Headers Required

```
#include <Etel3rdParty.h>
```

Library Needed

```
LIBRARY etel3rdparty.lib
```

Capability Required

```
Capability NetworkServices
```

Code

The following code assumes that you have a class for log text (CMyLog).

Define observer interface to be implemented by class you want it to be notified with events (if you like)

MPhoneReceiverObserver:

```
class MPhoneReceiverObserver
{
public:
 virtual void CallRinging() = 0;
 virtual void CallAnswered()=0;
};
```

CPhoneReceiver

```
class CPhoneReceiver : public CActive
{
public:
 // C++ constructor
 CPhoneReceiver(MPhoneReceiverObserver& aPhoneReceiverObserver);

 // Second-phase constructor
 void ConstructL();

 // Cancel and destroy
 ~CPhoneReceiver();

public:

 void StartL();

 //Open incoming call
 void OpenIncommingCall();

private: // From CActive
 // Handle completion
 void RunL();

 // How to cancel me
 void DoCancel();
```


(C) Copyright Nokia 2013. All rights reserved.

```
private:
 //Define object to CTelephone to manage calls
 CTelephony*    iTelephony;

 CTelephony::TCallInfoV1   iCurrentCallInfo;
 CTelephony::TCallInfoV1Pckg  iCurrentStatusPckg;

   //Define call ID to hold the ID of current call for further use (hangup, hold, resume)
 CTelephony::TCallId   iCallID;

 MPhoneReceiverObserver&   iPhoneReceiverObserver;


};
```

CPhoneReceiver implementation

```
CPhoneReceiver::CPhoneReceiver(MPhoneReceiverObserver& aPhoneReceiverObserver)
 : CActive(EPriorityStandard),
  iCurrentStatusPckg(iCurrentCallInfo),
  iPhoneReceiverObserver(aPhoneReceiverObserver)
{
}

void CPhoneReceiver::ConstructL()
{
 //Create new object to telephony
 iTelephony = CTelephony::NewL();

 CActiveScheduler::Add(this); // Add to scheduler
}

CPhoneReceiver::~CPhoneReceiver()
{
 Cancel(); // Cancel any request, if outstanding

 delete iTelephony;
}

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

void CPhoneReceiver::StartL()
{
 Cancel(); // Cancel any request, just to be sure

 //Notify of change in telephone line
 iTelephony->NotifyChange( iStatus, CTelephony::EVoiceLineStatusChange,
        iCurrentStatusPckg );

 SetActive(); // Tell scheduler a request is active
}

void CPhoneReceiver::RunL()
{
 if(iStatus.Int() == KErrNone)
 {
  //Get call status
  CTelephony::TCallStatus callStatus = iCurrentStatusPckg().iStatus;
  switch(callStatus)
  {
  case CTelephony::EStatusRinging:
   iPhoneReceiverObserver.CallRinging();
   break;

  case CTelephony::EStatusAnswering:
   iPhoneReceiverObserver.CallAnswered();
   break;
  }
 }
 else
 {
#ifdef __USE_PHONE_RECEIVER_LOG__
  CMyLog::OpenAndWriteToFile(_L("Error In status"));
#endif
 }

}


/**********************************************************************/
/* Define function to open incoming call                              */
/**********************************************************************/
void CPhoneReceiver::OpenIncommingCall()
{
#ifdef __USE_PHONE_RECEIVER_LOG__
 CMyLog::OpenAndWriteToFile(_L("OpenIncommingCall"));
#endif

 //Answer incoming call (Required Network services capability)
 iTelephony->AnswerIncomingCall(iStatus, iCallID);
 SetActive();
}
```

MyObserverClass

This can be you container class or any controller class depending on your program logic

```
class CMyObserverClass : public MPhoneReceiverObserver
{
public:
 void CallRinging();
 void CallAnswered();

private:
//Define object to phone receiver
 CPhoneReceiver* iPhoneReceiver;
```


(C) Copyright Nokia 2013. All rights reserved.

```
};
```

MyObserverClass Implementation

```
// -------------------------------------------------------
// CallRinging
// -------------------------------------------------------
void CMyObserverClass::CallRinging()
{
#ifdef __USE_PHONE_OBSERVER_LOG__
 CMyLog::OpenAndWriteToFile(_L("Call Ringing"));
#endif

 //Open line
 iPhoneReceiver->OpenIncommingCall();

}

/**********************************************************************/
/* This function is called when the phone call is answered              */
/**********************************************************************/
void CMyObserverClass::CallAnswered()
{
#ifdef __USE_PHONE_OBSERVER_LOG__
 CMyLog::OpenAndWriteToFile(_L("Call Answered"));
#endif
}
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.