



ID	CS000863	Creation date	March 28, 2008
Platform	S60 3rd Edition, MR S60 3rd Edition, FP1 S60 3rd Edition, FP2 Beta	Tested on devices	Nokia E61i Nokia E90 Communicator Nokia N95 8GB Nokia 6220 Classic
Category	Symbian C++	Subcategory	Telephony

Keywords (APIs, classes, methods, functions): CActive, CTelephony, CTelephony::TCallStatusV1, CTelephony::TCallStatusV1Pkg, CTelephony::NotifyChange(), CTelephony::TCallStatusV1::iStatus

Overview

Sometimes an application (such as a game) needs to be paused due to an incoming call. This snippet demonstrates how to observe incoming calls, thus enabling acting on them.

MMP file

The following libraries are required:

```
LIBRARY etel3rdparty.lib
```

Header file

```
#ifndef __TELOBSERVER_H_
#define __TELOBSERVER_H_

#include <e32base.h>
#include <Etel3rdParty.h>

class CTelObserver : public CActive
{
public:
    /**
     * Symbian OS default constructor
     */
    CTelObserver();
```

CS000863_-_Pausing_an_application_on_an_incoming_call

```
/**
 * 2nd phase constructor.
 */
static CTelObserver* NewL();

/**
 * 2nd phase constructor.
 */
static CTelObserver* NewLC();

/**
 * Destructor
 */
~CTelObserver();

/**
 * Starts observing for the incoming calls.
 */
void StartListening();

private:
/**
 * Symbian 2-phase constructor
 */
void ConstructL();

/**
 * From CActive
 */
void RunL();

/**
 * From CActive
 */
TInt RunError(TInt anError);

/**
 * From CActive
 */
void DoCancel();

private: // Data
    CTelephony* iTelephony;

    CTelephony::TCallStatusV1 iCallStatus;
    CTelephony::TCallStatusV1Pckg iCallStatusPkg;
};

#endif /*__TelObserver_H_*/
```

Source file

```
#include <e32base.h>
#include <Etel3rdParty.h>

#include "TelObserver.h"

/**
 * Constructor. Defines the priority for this active object.
 */
```

Header file

CS000863 - Pausing an application on an incoming call

```
CTelObserver::CTelObserver() :
    CActive(EPriorityStandard),
    iCallStatusPkg(iCallStatus)
{
}

/**
 * 2nd phase constructor.
 */
CTelObserver* CTelObserver::NewL()
{
    CTelObserver* self = CTelObserver::NewLC();
    CleanupStack::Pop(self);
    return self;
}

/**
 * 2nd phase constructor.
 */
CTelObserver* CTelObserver::NewLC()
{
    CTelObserver* self = new (ELeave) CTelObserver();
    CleanupStack::PushL(self);
    self->ConstructL();
    return self;
}

/**
 * 2nd phase constructor.
 */
void CTelObserver::ConstructL()
{
    iTelephony = CTelephony::NewL();

    CActiveScheduler::Add(this);
}

/**
 * Destructor.
 */
CTelObserver::~CTelObserver()
{
    Cancel();

    if (iTelephony)
        delete iTelephony;
}

/**
 * From CActive.
 */
void CTelObserver::RunL()
{
    {
        if (iStatus == KErrNone)
        {
            CTelephony::TCallStatus status = iCallStatus.iStatus;

            switch (status)
            {
                case CTelephony::EStatusRinging:
                {
                    // The phone is ringing. Pause the application.
                }
            }
        }
    }
}
```

CS000863 - Pausing an application on an incoming call

```
        // ...
        break;
    }
    default:
    {
        // Not interested in other events.
        break;
    }
}
// Start listening for the next call
StartListening();
}
}

/**
 * From CActive.
 */
TInt CTelObserver::RunError(TInt anError)
{
    return anError;
}

/**
 * From CActive.
 */
void CTelObserver::DoCancel()
{
    iTelephony->CancelAsync(CTelephony::EVoiceLineStatusChangeCancel);
}

/**
 * Starts observing for the incoming calls.
 */
void CTelObserver::StartListening()
{
    iTelephony->NotifyChange(iStatus, CTelephony::EVoiceLineStatusChange,
        iCallStatusPkg);
    SetActive();
}
}
```

Postconditions

When the phone is ringing, the application branches to `CTelephony::EStatusRinging` in `CTelObserver::RunL()`.