

## Detecting\_keypad\_lock/unlock\_status

Right now there is no observer/notifier pattern on the key lock API to notify the user of the keypad lock status, which means that to detect key lock activity the only choice is to keep polling the key lock api to check the status. This might not be a good idea as one can never be sure of the poll interval and it is not reliable either.

The following code snippet depicts a more reliable approach to detect the key pad lock status. Right now everytime the key pad is locked, the framework brings up a message like ?Keypad Locked? when the keypad is locked and ?Keypad unlocked? when it is unlocked, also if the user locks/unlocks the keypad manually then also the select dialog is popped up. The code below detects the popping up of those system dialogs which have a focus group id of either 3 or 9 and once they are detected it queries the key pad status, to see whats the status of the keypad.

Although the code is not 100% porting safe but it does the job right now till the time a notifier mechanism is implemented/published by Nokia.

Add following line to your .mmp file.

```
LIBRARY ws32.lib apgrfx.lib
```

**Add following lines in KeyPadLockStatusNotifier.h file.**

```
#include <W32STD.H>

class MKeyPadLockStatusObserver
{
public:
/**
    * Virtual function, implemented elsewhere
    */
virtual void KeyPadLockStatus(TBool aIslocked) = 0;
};

class CKeyPadLockStatusNotifier: public CActive
{
private:
    CKeyPadLockStatusNotifier(MKeyPadLockStatusObserver & aKeyPadLockStatusObserver);
void ConstructL();

public:
static CKeyPadLockStatusNotifier * NewL(MKeyPadLockStatusObserver & aKeyPadLockStatusObserver);
    ~ CKeyPadLockStatusNotifier();
void Start();

private:
// From CActive
void DoCancel();
void RunL();
    TInt iExtraError);

private:
    RWsSession iWs
    RWindowGroup ; iWindowGroup
    MKeyPadLockStatusObserver iKeyPadLockStatusObserver
};
```

## Detecting\_keypad\_lock/unlock\_status

### Add following lines in KeyPadLockStatusNotifier.cpp file.

```
#include <APGWGNAM.H>
#include <aknkeylock.h>

// User Include
#include "KeyPadLockStatusNotifier.h"

CKeyPadLockStatusNotifier::CKeyPadLockStatusNotifier (MKeyPadLockStatusObserver& aKeyPadLockStatusObserver,
    :CAActive(EPriorityIdle), iAppChangeObserver(aKeyPadLockStatusObserver)
{
}

CKeyPadLockStatusNotifier* CKeyPadLockStatusNotifier::NewL(MKeyPadLockStatusObserver& aKeyPadLockStatusObserver)
{
    CKeyPadLockStatusNotifier* pNotifier = new (ELeave)CKeyPadLockStatusNotifier(aKeyPadLockStatusObserver);
    CleanupStack::PushL(pNotifier);
    pNotifier->ConstructL();
    CleanupStack::Pop(pNotifier);
    return pNotifier;
}

void CKeyPadLockStatusNotifier::ConstructL()
{
    CActiveScheduler::Add(this);

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

    // Windows Group
    iWindowGroup = RWindowGroup(iWs);
    User::LeaveIfError(iWindowGroup.Construct(reinterpret_cast<TUint32> (&iWindowGroup), EFalse));
    iWindowGroup.SetOrdinalPosition(0, ECoeWinPriorityNeverAtFront);
    iWindowGroup.EnableReceiptOfFocus(EFalse);
    User::LeaveIfError(iWindowGroup.EnableFocusChangeEvents());
}

CKeyPadLockStatusNotifier::~CKeyPadLockStatusNotifier()
{
    Cancel();
    iWindowGroup.Close();
    iWs.Close();
}

void CKeyPadLockStatusNotifier::Start()
{
    iWs.EventReady(&iStatus);
    SetActive();
}

void CKeyPadLockStatusNotifier::DoCancel()
{
    iWs.EventReadyCancel();
}

void CKeyPadLockStatusNotifier::RunL()
{
    // Get the event from the window server session iWs
    TWsEvent wsEvent;
    GetEvent(wsEvent);
    // Get the event type: types are defined in TEventCode
    switch (wsEvent.Type())
    {

```

## Detecting\_keypad\_lock/unlock\_status

```
// Window-group related event types
case EEventFocusGroupChanged:
{
    TInt wgid = WsGetFocusWindowGroup();
    if(wgid == 3 || wgid == 9)
    {
        RAknKeylock2 keyLock;
        TInt err;
        (err, keyLock.Connect()); TRAP
    if(err == KErrNone)
    {
        TBool locked = keyLock.IsKeyLockEnabled(TKeyPadLockStatusKeyPadLockStatusObserver);
        keyLock.Close();
    }
    }
    break;
}

TInt CKeyPadLockStatusNotifier::RunError(TInt aError)
{
    return KErrNone;
}
```

### Add following lines in YourClass.h file.

```
class CYourClass:public CBase, public MKeyPadLockStatusObserver
{
public:
    void KeyPadLockStatus(TBool aIslocked);
public:
    CKeyPadLockStatusNotifier* iKeyPadLockStatusNotifier ;
};
```

### Add following lines in YourClass.cpp file.

```
// Register for Ws Changes to be notified
iKeyPadLockStatusNotifier = CKeyPadLockStatusNotifier::NewL(*this);

void CYourClass:: KeyPadLockStatus (TBool aIslocked)
{
    // Do something
}
```

The keypad lock status is returned to the user, where they can do the processing based on the status of the same.

## Link

[KIS001084 - Detecting keypad lock/unlock events not possible](#)

[KIS000831 - Querying Keypad status using GetIndicatorPayload\(\) does not work with custom screensaver](#)

Detecting\_keypad\_lock/unlock\_status

**Added by - Mayank on 18/06/2009**