

ID	TSS000468	Creation date	November 22, 2006
Platform	S60 2nd Edition, S60 2nd Edition FP1, FP2, and FP3 S60 3rd Edition, S60 3rd Edition FP1	Devices	
Category	Symbian C++	Subcategory	

Keywords (APIs, classes, methods, functions):

Overview

Receiving notifications for installation events

Description

Applications may have the need to react to software installation events. One such scenario is when an application needs to launch the installation of a SIS file, monitor its progress, and take action upon a successful installation.

Solution

Launching the installation of a SIS file can be done with RApalsSession API. In the following example code, iApaSession and iFs are already open handles to RApalsSession and RFs. Also, it is assumed that the code belongs to a class that derives from CActive.

```
TThreadId threadId;
User::LeaveIfError(iApaSession.StartDocument(instSource, threadId));
// logon to the midlet installer thread
// iInstThread is the SW installer thread (of type RThread)
User::LeaveIfError(iInstThread.Open(threadId));
iInstThread.Logon(iStatus);
SetActive();
```

When the installation ends - successfully or otherwise, RunL() function (from CActive) will be called. To find out whether the installation was successful, RApalsSession::GetAppInfo() can be used:

```
TApaAppInfo appInfo;
if (aApaSession.GetAppInfo(appInfo, KInstalledAppUid) == KErrNone)
{
    // application was installed successfully
}
```

```

    }

```

Note however, that the list of installed applications is not updated immediately, and calling `GetAppInfo()` too early may return `KErrNotFound` even though the application was installed. To solve this on S60 2nd Edition, it may be helpful to request change notifications on the data file where application information is stored:

```

_LIT(KApplicationsDataFilePath, "C:\\System\\Data\\Applications.dat");
iFs.NotifyChange(ENotifyWrite, iStatus, KApplicationsDataFilePath);

```

S60 3rd Edition

S60 3rd Edition provides `Publish` and `Subscribe` keys that makes it easy to monitor the installer status, and retrieve the UID of the most recently installed application.

```

#include <e32property.h>

```

```

#include <sacls.h>

```

```

TInt instStatus;

```

```

User::LeaveIfError( RProperty::Get( KUidSystemCategory, KSAUidSoftwareInstallKeyValue, instStatus )
);

```

To get notifications of changes to this key, use `Attach()` and `Subscribe()` functions from `RProperty`. The returned value contains bitwise-combined flags from the following two enumerations:

```

enum TInstOp

```

```

{

```

```

    EInstOpNone = 0x00000000,

```

```

    EInstOpInstall = 0x00000001,

```

```

    EInstOpUninstall = 0x00000002,

```

```

    EInstOpRestore = 0x00000004

```

```

};

```

```

enum TInstOpStatus

```

```

{

```

```

    EInstOpStatusNone = 0x00000000,

```

```

    EInstOpStatusSuccess = 0x00000100,

```

```

    EInstOpStatusAborted = 0x00000200

```

```

};

```

The UID of the most recently installed or updated application can be retrieved as follows:

```

TInt appUid;

```

```

User::LeaveIfError(RProperty::Get(KUidSystemCategory, KUidSwiLatestInstallation, appUid));

```