



|                 |                      |                          |              |
|-----------------|----------------------|--------------------------|--------------|
| <b>ID</b>       | CS000986             | <b>Creation date</b>     | May 28, 2008 |
| <b>Platform</b> | S60 3rd Edition, FP1 | <b>Tested on devices</b> | Nokia N93    |
| <b>Category</b> | Symbian C++          | <b>Subcategory</b>       | Files/Data   |

**Keywords (APIs, classes, methods, functions):** CFileStore, RStoreReadStream, CDirectFileStore, CScheduledTask, RScheduler, CFileStore::Root(), RStoreReadStream::OpenLC(), RStoreReadStream::ReadInt32L(), CDirectFileStore::FromLC(), CScheduledTask::Info(), CScheduledTask::Data(), CScheduledTask::ValidUntil(), RScheduler::Connect(), RScheduler::Register(), RScheduler::Close()

## Overview

This code snippet shows how a task handler EXE program can be created and registered when the class `RScheduler` is used to schedule Task Scheduler tasks. The client has to be registered with the scheduler before any tasks can be scheduled. The registration method is called `Register()` and it takes two parameters, the filename of the program to execute the tasks and the priority which is relative to other clients.

This snippet can be self-signed.

## MMP file

The following libraries are required:

```
LIBRARY          euser.lib
LIBRARY          efsrv.lib
LIBRARY          estor.lib
LIBRARY          schsvr.lib
```

## Source file (ExampleTaskHandler.exe)

```
#include <schinfo.h>
#include <schtask.h>
#include <s32file.h>
```

## CS000986\_-\_Creating\_and\_registering\_a\_task\_handler\_with\_RScheduler

```
static void DoRunTaskL(RFile& aTaskFile)
{
    TInt taskCount(0);
    CFileStore* fileStore;
    RStoreReadStream readStream;

    fileStore = CDirectFileStore::FromLC(aTaskFile);
    readStream.OpenLC(*fileStore, fileStore->Root());

    //from schtask.h :
    //The root stream of the direct file store contains a 32-bit value, followed
    //by the external representations of one or more CScheduledTask objects. The
    //32-bit value is interpreted as a TInt32 and contains the number of CScheduledTask
    //objects that follow in the stream.
    taskCount = readStream.ReadInt32L();

    for (TInt index = 0; index < taskCount; index++)
    {
        CScheduledTask* task = CScheduledTask::NewLC(readStream);

        // Do something with the task info...
        //task->Info().iName;
        //task->Info().iTaskId;
        //task->ValidUntil();
        //HBufC* data = const_cast<HBufC*>(&(task->Data()));

        CleanupStack::PopAndDestroy(task);
    }

    CleanupStack::PopAndDestroy(2); // fileStore, readStream
}

LOCAL_D TInt RunTask()
{
    TInt err(KErrNoMemory);
    CTrapCleanup* cleanup=CTrapCleanup::New();

    if (cleanup)
    {
        RFile file;

        //from schtask.h :
        //The registered program can use the RFile::AdoptFromCreator API in conjunction
        //with the APIs provided by this class to access the scheduled task file store
        err = file.AdoptFromCreator(TScheduledTaskFile::FsHandleIndex(),
                                   TScheduledTaskFile::FileHandleIndex());

        if (err != KErrNone)
        {
            return err;
        }

        TRAP(err, DoRunTaskL(file));

        file.Close();

        delete cleanup;
    }

    return err;
}

GLDEF_C TInt E32Main()
```

```
{
    return RunTask();
}
```

## Header file

```
#ifndef __SCHEDULEREXAMPLEAPPUI_H__
#define __SCHEDULEREXAMPLEAPPUI_H__

#include <csch_cli.h> // RScheduler

class CSchedulerExampleAppUi : public CAknAppUi
{
    //...
private:
    RScheduler iScheduler;

};

#endif // __SCHEDULEREXAMPLEAPPUI_H__
```

## Source file

```
void CSchedulerExampleAppUi::ConstructL()
{
    //...
    User::LeaveIfError(iScheduler.Connect());

    _LIT(KExampleTaskHandlerExe, "ExampleTaskHandler.exe");
    TFileName exampleHandler(KExampleTaskHandlerExe);

    User::LeaveIfError(iScheduler.Register(exampleHandler, CActive::EPriorityStandard));
}

CSchedulerExampleAppUi::~CSchedulerExampleAppUi()
{
    //...
    iScheduler.Close();
}
```

## Postconditions

The client is registered with the scheduler and ready to schedule Task Scheduler tasks. The program ExampleTaskHandler.exe is responsible for running scheduled tasks.

## See also

- [CS000987 - Creating persistent and transient schedules with RScheduler](#)
- [CS000988 - Creating a condition-based schedule with RScheduler](#)
- [CS000989 - Getting schedule and task info using RScheduler](#)
- [CS000990 - Getting schedule and task count using RScheduler](#)
- [CS000991 - Editing a schedule using RScheduler](#)
- [CS000992 - Deleting schedules and tasks using RScheduler](#)