



<b>ID</b>	CS000990	<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):** `RScheduler`, `TSchedulerItemRef`, `TScheduleEntryInfo2`, `TTaskSchedulerCondition`, `TTaskInfo`, `TTsTime`, `TScheduleState2`, `RScheduler::Connect()`, `RScheduler::Register()`, `RScheduler::Close()`, `RScheduler::GetScheduleRefsL()`, `RScheduler::GetScheduleL()`

## Overview

This code snippet implements three simple helper methods to get a count of schedules, scheduled condition-based tasks, and scheduled time-based tasks. The `RScheduler` method `GetScheduleRefsL()` is used to get the count of schedules. The filter parameter of this method can be used to include all schedules (`EAllSchedules`) or only pending (`EPendingSchedules`) schedules to count.

The `RScheduler` method `GetScheduleL()` is used to get task count of the schedule by a given ID. This example implements separate methods for time-based and condition-based schedules. All these helper methods return -1 in error situations.

This snippet can be self-signed.

## MMP file

The following libraries are required:

```
LIBRARY          schsvr.lib
```

## Preconditions

`ExampleTaskHandler.exe` must be created and time-based and condition-based schedules with example tasks must be scheduled before this code snippet can be executed. See code snippets [CS000986 - Creating and registering a task handler with RScheduler](#), [CS000987 - Creating persistent and transient schedules with RScheduler](#), and [CS000988 - Creating a condition-based schedule with RScheduler](#) for more information.

## Resource files

.rsrc

```
RESOURCE MENU_PANE r_schedulerexample_menu
{
    items =
    {
        //...
        MENU_ITEM {command = EGetCountExample;          txt = "GetCountExample";},
        //...
        MENU_ITEM {command = EAknSoftkeyExit;          txt = "Exit";}
    };
}
```

.hrh

```
enum TSchedulerExampleIds
{
    //...
    EGetCountExample,
    //...
};
```

## Header file

```
#ifndef __SCHEDULEREXAMPLEAPPUI_H__
#define __SCHEDULEREXAMPLEAPPUI_H__

#include <csch_cli.h> // RScheduler
#include <schinfo.h> // TSchedulerItemRef, TTaskInfo...

class CSchedulerExampleAppUi : public CAknAppUi
{
    //...
public:
    void HandleCommandL(TInt aCommand);
    //...
private:
    TInt GetConditionBasedScheduleTaskCountL(RScheduler& aScheduler,
                                             TInt aScheduleId,
                                             TInt& aTaskCount);

    TInt GetTimeBasedScheduleTaskCountL(RScheduler& aScheduler,
                                         TInt aScheduleId,
                                         TInt& aTaskCount);

    TInt GetScheduleCountL(TScheduleFilter aFilter,
                          RScheduler& aScheduler,
                          TInt& aScheduleCount);
    //...
private:
    RScheduler iScheduler;

    TSchedulerItemRef iPersistentScheduleHandle;
    TSchedulerItemRef iConditionScheduleHandle;
```

```
};

#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();
}

void CSchedulerExampleAppUi::HandleCommandL(TInt aCommand)
{
    TBuf<100> Text1; //first line of dialog text
    TBuf<100> Text2; //second line of dialog text

    switch(aCommand)
    {
        case EEikCmdExit:
        case EAknSoftkeyExit:
            Exit();
            break;
        //...
        case EGetCountExample:
        {
            TInt err(KErrNone);
            TInt scheduleCount=-1;
            TInt timeBasedScheduleTaskCount=-1;
            TInt conditionBasedScheduleTaskCount=-1;

            err = GetScheduleCountL(EAllSchedules, iScheduler, scheduleCount);

            err = GetTimeBasedScheduleTaskCountL(iScheduler,
            iPersistentScheduleHandle.iHandle, timeBasedScheduleTaskCount );

            err = GetConditionBasedScheduleTaskCountL(iScheduler,
            iConditionScheduleHandle.iHandle, conditionBasedScheduleTaskCount );

            Text1.Append(_L("ScheduleCount:"));
            Text1.AppendNum(scheduleCount);

            Text2.Append(_L("TimeBasedScheduleTaskCount:"));
            Text2.AppendNum(timeBasedScheduleTaskCount);
            Text2.Append(_L("\nConditionBasedScheduleTaskCount:"));
        }
    }
}
```

## CS000990\_-\_Getting\_schedule\_and\_task\_count\_using\_RScheduler

```
Text2.AppendNum(conditionBasedScheduleTaskCount);

CEikonEnv::Static()->InfoWinL(Text1, Text2);
}
break;

default:
    //Panic(ESchedulerExampleUi);
    break;
}
}

TInt CSchedulerExampleAppUi::GetConditionBasedScheduleTaskCountL(RScheduler& aScheduler,
                                                                TInt aScheduleId,
                                                                TInt& aTaskCount)
{
    TInt ret(KErrNone);

    CArrayFixFlat<TTaskInfo>* tasks =
new (ELeave) CArrayFixFlat<TTaskInfo>(1);
    CleanupStack::PushL(tasks);

    TTsTime time;
    TScheduleState2 state;
    CArrayFixFlat<TTaskSchedulerCondition>* conditions =
new (ELeave) CArrayFixFlat<TTaskSchedulerCondition>(1);
    CleanupStack::PushL(conditions);

    tasks->Reset();
    ret = aScheduler.GetScheduleL(aScheduleId,
                                state,
                                *conditions,
                                time,
                                *tasks);
    CleanupStack::PopAndDestroy(conditions);

    if(ret == KErrNone)
        aTaskCount = tasks->Count();
    else
        aTaskCount = -1;

    CleanupStack::PopAndDestroy(tasks);

    return ret;
}

TInt CSchedulerExampleAppUi::GetTimeBasedScheduleTaskCountL(RScheduler& aScheduler,
                                                            TInt aScheduleId,
                                                            TInt& aTaskCount)
{
    TInt ret(KErrNone);

    CArrayFixFlat<TTaskInfo>* tasks = new (ELeave) CArrayFixFlat<TTaskInfo>(1);
    CleanupStack::PushL(tasks);

    TTsTime time;
    TScheduleState2 state;
    CArrayFixFlat<TScheduleEntryInfo2>* entries =
new (ELeave) CArrayFixFlat<TScheduleEntryInfo2>(1);
    CleanupStack::PushL(entries);
```

## CS000990\_-\_Getting\_schedule\_and\_task\_count\_using\_RScheduler

```
tasks->Reset();
ret = aScheduler.GetScheduleL(aScheduleId,
                             state,
                             *entries,
                             *tasks,
                             time);

CleanupStack::PopAndDestroy(entries);

if(ret == KErrNone)
    aTaskCount = tasks->Count();
else
    aTaskCount = -1;

CleanupStack::PopAndDestroy(tasks);

return ret;
}

TInt CSchedulerExampleAppUi::GetScheduleCountL(TScheduleFilter aFilter,
                                               RScheduler& aScheduler,
                                               TInt& aScheduleCount)
{
    TInt ret(KErrNone);

    CArrayFixFlat<TSchedulerItemRef>* items =
new (ELeave) CArrayFixFlat<TSchedulerItemRef> (1);
    CleanupStack::PushL(items);

    ret = aScheduler.GetScheduleRefsL(*items, aFilter);

    if(ret == KErrNone)
        aScheduleCount = items->Count();
    else
        aScheduleCount = -1;

    CleanupStack::PopAndDestroy(items);

    return ret;
}
```

## Postconditions

The count of schedules, time-based schedule tasks, and condition-based schedule tasks are displayed to the user with info dialogs.

## See also

- [CS000986 - Creating and registering a task handler with RScheduler](#)
- [CS000987 - Creating persistent and transient schedules with RScheduler](#)
- [CS000988 - Creating a condition-based schedule with RScheduler](#)

## CS000990\_-\_Getting\_schedule\_and\_task\_count\_using\_RScheduler

- [CS000989 - Getting schedule and task info using RScheduler](#)
- [CS000991 - Editing a schedule using RScheduler](#)
- [CS000992 - Deleting schedules and tasks using RScheduler](#)