

<b>ID</b>	TSS001441	<b>Creation date</b>	June 23, 2009
<b>Platform</b>	S60 3rd Edition, FP1 S60 3rd Edition, FP2	<b>Devices</b>	Tested on Nokia N95
<b>Category</b>	Symbian C++	<b>Subcategory</b>	Signing and Certification

**Keywords (APIs, classes, methods, functions):** CUnifiedCertStore

## Description

The `CUnifiedCertStore` class provides a certificate store whose contents are the sum of the contents of all certificate store implementations on the device. It is intended as the single point of access for clients wishing to use certificate stores.

## Solution

The following sample code shows how to retrieve the certificates present on the device.

### MyClass.h

```
#include <unifiedcertstore.h>      // link against certstore.lib
#include <f32file.h>               // link against efsrv.lib
#include <ccertattributefilter.h> // link against ctframework.lib

// Declaration of CMyClass

class CMyClass : public CActive
{
public:

    enum TState
    {
        EGetCerts,
        EInitialize
    };

    static CMyClass* NewL();
    ~CMyClass();

    void InitL();
    void DoListL();
};
```

## TSS001441\_-\_Listing\_the\_certificates\_on\_the\_device\_using\_unified\_certificate\_store\_API

```
void PopulateListL();

protected: // From CActive
    virtual void DoCancel();
    virtual void RunL();

private: // Private constructors
    CMyClass();
    void ConstructL();

private: // Data

    CUnifiedCertStore*      iCertStore;
    RMPointerArray<CCTCertInfo> iCerts;
    CActiveSchedulerWait    iWait;

    TState                  iCertCmd;
    RFs                      iFs;
};
```

### MyClass.cpp

```
#include "MyClass.h"

CMyClass* CMyClass::NewL()
{
    CMyClass* self = new (ELeave) CMyClass;
    CleanupStack::PushL(self);
    self->ConstructL();
    CleanupStack::Pop(); // self
    return self;
}

CMyClass::CMyClass()
: CActive(CActive::EPriorityStandard)
{
    CActiveScheduler::Add( this );
}

CMyClass::~CMyClass()
{
    Cancel();

    delete iCertStore;
    iFs.Close();
    iCerts.Close();
}

void CMyClass::ConstructL()
{
    User::LeaveIfError( iFs.Connect() );
    iCertStore = CUnifiedCertStore::NewL( iFs, EFalse );
    InitL();
}
```

## TSS001441\_-\_Listing\_the\_certificates\_on\_the\_device\_using\_unified\_certificate\_store\_API

```
void CMyClass::InitL()
{
    if( IsActive() )
    {
        return;
    }

    iCertStore->Initialize( iStatus );
    SetActive();
    iCertCmd = EInitialize;
    iWait.Start();
    // RunL called when this completes
}

void CMyClass::DoListL()
{
    CCertAttributeFilter* certFilter = CCertAttributeFilter::NewLC();
    certFilter->SetFormat( EX509Certificate );
    iCertStore->List(iCerts, *certFilter, iStatus);
    SetActive();
    iCertCmd = EGetCerts;
    iWait.Start();
    // RunL called when this completes

    CleanupStack::PopAndDestroy(); // certFilter
}

void CMyClass::RunL()
{
    {
        iWait.AsyncStop();
        if(iStatus == KErrNone)
        {
            switch(iCertCmd)
            {
                {
                    case EInitialize:
                    {
                        DoListL();
                        PopulateListL();
                        break;
                    }
                    default:
                    break;
                }
            }
        }
    }
}

void CMyClass::DoCancel()
{
    {
        switch(iCertCmd)
        {
            {
                case EInitialize:
                {
                    iCertStore->CancelInitialize();
                    break;
                }
                case EGetCerts:
                {
                    iCertStore->CancelList();
                }
            }
        }
    }
}
```

## TSS001441\_-\_Listing\_the\_certificates\_on\_the\_device\_using\_unified\_certificate\_store\_API

```
        break;
    }
    default:
        break;
    }
}

void CMyClass::PopulateListL()
{
    for( TInt i = 0; i < iCerts.Count(); i++ )
    {
        TBuf<KMaxCertLabelLength>    iCertLabel;
        iCertLabel.Copy( iCerts[i]->Label() );
        // ... process each cert label here
    }
}
```

**Required capabilities:** ReadDeviceData