Using RProperty for subscribing

Overview

This code snippet shows how the class RProperty can be used to subscribe user-defined properties published by Symbian's IPC Publish and Subscribe mechanism. The following steps are needed to subscribe and retrieve values of the updated properties.

- Register interest by attaching to the property with `RProperty::Attach()` and call `RProperty::Subscribe()`.
- Get notified via a signal to the `TRequestStatus` to complete the `RProperty::Subscribe()`.
- Retrieve the value of the updated property by calling `RProperty::Get()`.
- Resubmit a request for notification of changes to the property by calling `RProperty::Subscribe()`.
- Finally unsubscribe by calling `RProperty::Cancel()` and close the handle `RProperty::Close()`.

The code snippet Archived:Using RProperty for publishing shows how these user-defined properties can be published. The property definitions are shared between a publisher and a subscriber through the common header file `ExampleProperties.h`.

```
#ifndef EXAMPLEPROPERTIES_H_
#define EXAMPLEPROPERTIES_H_

const TInt KMaxStrLen = 10;
const TUid KExampleProperty = {0xED1917A8};
enum TExamplePropertyKeys  { EIntProperty, EStrProperty };

#endif /*EXAMPLEPROPERTIES_H_*/
```

This snippet can be self-signed.

MMP file

The following libraries are required:

```
LIBRARY euser.lib
```

Preconditions

The publisher or the subscriber thread can be the one to define properties by calling `RProperty::Define()`. The user-defined properties persist in the kernel until the operating system reboots or the properties are deleted.

```
TInt ret=RProperty::Define(KExampleProperty,EIntProperty,RProperty::EInt);
if (ret != KErrAlreadyExists)
  {
  User::LeaveIfError(ret);
  }

ret= RProperty::Define(KExampleProperty,EStrProperty,RProperty::EByteArray,KMaxStrLen);
if (ret != KErrAlreadyExists)
  {
  User::LeaveIfError(ret);
  }
```

Header file

```
#ifndef EXAMPLESUBSCRIBER_H_
#define EXAMPLESUBSCRIBER_H_

//  INCLUDES
#include <e32base.h>
#include <e32property.h>

#include "exampleproperties.h"


class MExampleSubscriberObserver
{
public:
  virtual void IntPropertyUpdatedL(TInt aValue) = 0;
  virtual void StrPropertyUpdatedL(TDes& aValue) = 0;
};


class CExampleSubscriber : public CActive
        {
        enum {EPriority=0};
public:
        static CExampleSubscriber* NewL( const TUid aUid,
                                         const TUint32 aKey,
                                         MExampleSubscriberObserver& aNotifier );
        virtual ~CExampleSubscriber();

private:
        CExampleSubscriber( const TUid aUid, const TUint32 aKey,
```


(C) Copyright Nokia 2013. All rights reserved.

```
                                MExampleSubscriberObserver& aNotifier );
        void ConstructL();
        void RunL();
        void DoCancel();
private:
      RProperty iProperty;
        const TUid iUid;
        const TUint32 iKey;
        MExampleSubscriberObserver&  iObserver;
        };

#endif /*EXAMPLESUBSCRIBER_H_*/
```

Source file

```
#include "ExampleSubscriber.h"

CExampleSubscriber::CExampleSubscriber(
        const TUid aUid, const TUint32 aKey,
        MExampleSubscriberObserver& aObserver )
            : CActive(EPriority), iUid( aUid ),
              iKey( aKey ), iObserver( aObserver)
        {
        }

CExampleSubscriber* CExampleSubscriber::NewL( const TUid aUid, const TUint32 aKey,
                              MExampleSubscriberObserver& aObserver )
        {
        CExampleSubscriber* self=
            new(ELeave) CExampleSubscriber( aUid, aKey, aObserver );
        CleanupStack::PushL(self);
        self->ConstructL();
        CleanupStack::Pop(self);
        return self;
        }

void CExampleSubscriber::ConstructL()
        {
        User::LeaveIfError( iProperty.Attach( iUid, iKey ) );
        CActiveScheduler::Add(this);
        // initial subscription and process current property value
        RunL();
        }

CExampleSubscriber::~CExampleSubscriber()
        {
        Cancel();
        iProperty.Close();
        }

void CExampleSubscriber::DoCancel()
        {
        iProperty.Cancel();
        }

void CExampleSubscriber::RunL()
  {
    // resubscribe before processing new value to prevent missing updates
    iProperty.Subscribe( iStatus );
    SetActive();

    TInt intValue;
    TBuf<KMaxStrLen> strValue;

    switch( iKey )
        {
        case EIntProperty:
          // Int property updated, get new value
          //see also: RProperty::Get(KExampleProperty,EIntProperty,intValue);
          if( iProperty.Get( intValue ) == KErrNotFound )
            {
              // Int property deleted, do necessary actions here...
                 }
             else
              {
              // use new value ...
              iObserver.IntPropertyUpdatedL(intValue);
              }
          break;
        case EStrProperty:
          // Int property updated, get new value
          //see also: RProperty::Get(KExampleProperty,EStrProperty,strValue);
          if( iProperty.Get( strValue ) == KErrNotFound )
            {
              // Str property deleted, do necessary actions here...
              }
          else
            {
              // use new value ...
              iObserver.StrPropertyUpdatedL(strValue);
              }
          break;

        default:
            break;
        }
  }

//  End of File
```

Using CExampleSubscriber class

- In the header file:

```
#include "examplesubscriber.h"


class CPropertyExampleAppUi : public CAknAppUi, MExampleSubscriberObserver
```


(C) Copyright Nokia 2013. All rights reserved.

```
  {
//...
public:
  void IntPropertyUpdatedL(TInt aValue);
  void StrPropertyUpdatedL(TDes& aValue);
//...
  CExampleSubscriber* iIntSubscriber;
  CExampleSubscriber* iStrSubscriber;
        };
```

- In the source file:

```
void CPropertyExampleAppUi::ConstructL()
  {
//...
   iIntSubscriber = CExampleSubscriber::NewL(KExampleProperty,EIntProperty,*this);
    iStrSubscriber = CExampleSubscriber::NewL(KExampleProperty,EStrProperty,*this);
  }

CPropertyExampleAppUi::~CPropertyExampleAppUi()
  {
//...
  delete iIntSubscriber;
  delete iStrSubscriber;
  }

void CPropertyExampleAppUi::IntPropertyUpdatedL(TInt aValue)
        {
  //do something with the updated property
        }

void CPropertyExampleAppUi::StrPropertyUpdatedL(TDes& aValue)
        {
  //do something with the updated property
  }
```

Postconditions

Two example properties are subscribed using the class RProperty.

If the subscriber thread defined the properties, it is also capable of deleting properties by using `RProperty::Delete()` as follows:

```
TInt err(KErrNone);

err = RProperty::Delete(KExampleProperty,EIntProperty);
if (err != KErrNotFound)
  {
  User::LeaveIfError(err);
  }
err = RProperty::Delete(KExampleProperty,EStrProperty);
if (err != KErrNotFound)
  {
  User::LeaveIfError(err);
  }
```

When properties are deleted, any outstanding subscriptions will be completed with KErrNotFound.

See also

Archived:Using RProperty for publishing

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.