Using multiple positioning methods

The S60 C++ Location Acquisition API does not currently support automatically prioritizing positioning methods based on the developer's needs. For example when opening an RPositioner subsession for the default positioning module, the network based PSY will not be used if any other PSYs have been selected as the active positioning methods.

I.e. a call like this will never revert to the network based PSY if other positioning methods are also activated.

```
iPositioner.Open(iPosServer);
```

To use multiple positioning methods in your application you need to create separate positioning requestors for the methods you want to use, for example a requestor for the network based PSY and a requestor for the default positioning method.

One approach is also to first check if the network based PSY can give you a location estimate and then revert back to the default PSY. This approach is highlighted below, by documenting the changes needed to the S60 SDK LBS Reference Application, to support co-existence of network based PSY and the default PSY.

Introduce a state variable to indicate we're trying to get the network based location estimate.

```
// LbsPositionRequestor.h
TBool iGettingNetBasedPosition;
```

In CLbsPositionRequestor::DoInitialiseL(), open the RPositioner to use the network based PSY instead of the default one.

Note: The code snipped below identifies the A-GPS module based on its Technology Type. It can also be done using module ID (TPositionModuleInfo::ModuleId()) or capabilities (TPositionModuleInfo::Capabilities()) or module name in your identification logic.Please note that it is not recommended to use module's name because it is localized and the code snippet would fail when a language other than English is used in device.

```
TUint numOfModules = 0;
User::LeaveIfError( iPosServer.GetNumModules( numOfModules ) );

   for(TInt i = 0 ; i < numOfModules ; i++)
    {
        // read current module info
        TPositionModuleInfo modInfo;
        TInt error = iPosServer.GetModuleInfoByIndex( i, modInfo );

        if(modInfo.IsAvailable() && (modInfo.TechnologyType() ==
TPositionModuleInfo::ETechnologyNetwork))
       {
         // Open subsession to the position server
         TInt err = iPositioner.Open(iPosServer,modInfo.ModuleId());
         break;
        }
     }
```

In CLbsPositionRequestor::PositionUpdatedL(), set iGettingNetBasedPosition to ETrue, when you have received the info about last known position.

Implement a funnction to revert back to the default PSY, once the first location estimate from the network based PSY has been received.

```
/**
```


(C) Copyright Nokia 2014. All rights reserved.

```
* Tears down the network based positioner
* after an initial NET PSY fix or a timeout/request error
*/
void CLbsPositionRequestor::RevertToDefaultPSYL()
 {
 iPositionListener.ShowMessageL(_L("reverting to default PSY"));
 iGettingNetBasedPosition = EFalse;

 iPositioner.Close();
 iPositioner.Open( iPosServer );
 // Set position requestor
 iPositioner.SetRequestor( CRequestor::ERequestorService ,
             CRequestor::EFormatApplication, KRequestor );
 }
```

Revert to the default PSY after getting a fix from the network based PSY. Here's the complete CLbsPositionRequestor::PositionUpdatedL() function.

```
// ---------------------------------------------------------------------------
// CLbsPositionRequestor::PositionUpdatedL
// Pre process the position information
// ---------------------------------------------------------------------------
//
void CLbsPositionRequestor::PositionUpdatedL()
    {
    //If we are getting the last known position
    if ( iGettingLastknownPosition )
        {
        // Set the module name to 'Unknown' because last
        // known position has no module name
        iModuleName = KUnknown;

        // Send position information to registered listener
        iPositionListener.PositionInfoUpdatedL(*iPosInfoBase,iModuleName);

        //Change the data class type
        iPosInfoBase = &iSatelliteInfo;

        //
        // try to get net based position next
        iGettingNetBasedPosition = ETrue;
        //

        iPositionListener.ShowMessageL( _L("going for net psy") );
        // Request next position
        iPositioner.NotifyPositionUpdate( *iPosInfoBase, iStatus );

        // Set this object active
        SetActive();
        }
    else
        {
        // Check if the id of the used PSY is 0
        if ( 0 == iUsedPsy.iUid)
            {
            // Set the id of the currently used PSY
```


(C) Copyright Nokia 2014. All rights reserved.

```
            iUsedPsy = iPosInfoBase->ModuleId();
            }
        // Check if the position module has changed
        else if ( iPosInfoBase->ModuleId() != iUsedPsy )
            {
            // Set the id of the currently used PSY
            iUsedPsy = iPosInfoBase->ModuleId();

            //Position module info of new module
            TPositionModuleInfo moduleInfo;

            // Get module info
            iPosServer.GetModuleInfoById(iUsedPsy,moduleInfo);

            // Get classes supported
            TInt32 moduleInfoFamily = moduleInfo.ClassesSupported(EPositionInfoFamily);

            iPosInfoBase = &iSatelliteInfo;

            // Check if the new module supports
            // TPositionSatelliteInfo class
            if ( EPositionSatelliteInfoClass & moduleInfoFamily )
                {
                // Set info base to satellite info
                iPosInfoBase = &iSatelliteInfo;
                }
            // The position module must support atleast
            // TPositionInfo class
            else
                {
                // Set info base to position info
                iPosInfoBase = &iPositionInfo;
                }
            }

        // Process the position information
        // and request next position

        // Get module name
        GetModuleName(iUsedPsy);

        // Send position information to registered listener
        iPositionListener.PositionInfoUpdatedL(*iPosInfoBase,iModuleName);

        //
        // --------------------
        if( iGettingNetBasedPosition )
         {
         RevertToDefaultPSYL();
         }

     // Request next position
        iPositioner.NotifyPositionUpdate( *iPosInfoBase, iStatus );

        // Set this object active
        SetActive();
        //
```


(C) Copyright Nokia 2014. All rights reserved.

```
        // --------------------

        }
    }
```

This should be it for the minimal set of changes needed. Please pay attention to the error handling. The changes above expect that network based PSY is active in S60 device Settings application, and that no errors occur. I.e. the code is for demo purposes only.

(C) Copyright Nokia 2014. All rights reserved.