S60 Sensor Framework

02 Nov 2008

The S60 Sensor Framework is a platform-level set of APIs meant to provide a consistent method of accessing the sensor hardware that is becoming a constant presence on S60 phones. For a brief overview of the sensor API offerings and the fragmentation that existed before the release of this framework, see the Sensor category page.

The S60 Sensor Framework was introduced in S60 5th Edition, but is also backported to S60 3rd Edition, Feature Pack 2 for some mobile devices, as well as the Nokia E66 device, which is an S60 3rd Edition, Feature Pack 1 device with sensor APIs based on the S60 Sensor Framework.

Note that the old sensor API plug-ins are not compatible with the S60 Sensor Framework. In addition, applications created with Sensor FW do not work in S60 3rd Edition, FP1 devices other than the Nokia E66 mobile device. If you wish to provide just one installation file, you can pack the sensor API parts into an ECom plug-in, build one file with the S60 5th Edition SDK and another with the S60 3rd Edition MR SDK, and then load the correct one during runtime.

General

The S60 Sensor Framework is an expandable system based on a central framework that handles the sensor data provider plug-ins under it. This means that different handsets can have different sets of sensors, and that some sensor data providers can be installed later onto the devices.

Each application can utilise multiple sensor data providers at the same time, and the same data provider can be used by multiple applications at any given time.

Also, each physical hardware sensor can be used by multiple sensor data providers, for example, the accelerometer sensor is also used for the double tapping sensor.

S60 Sensor Framework APIs

There are three main APIs that can be used in the S60 Sensor Framework:

- Sensor Plug-in API
- Channel Finder API
- Sensor Channel API

The Sensor Plug-in API is an internal S60 API that is not included in the public SDK. It is intended to be used by S60 licensees to provide more data provider plug-ins.

The Channel Finder API is used to enumerate the channels available on a device, and the Sensor Channel API is used to communicate with the sensor data provider. The latter has the following functionalities:

- Methods for opening and closing the channel;
- Methods for starting and stopping data listening from the sensor data provider;
- Methods for setting and reading properties set for the sensor data provider channel.

Utilising the Channel Finder API

The Channel Finder API is used to retrieve all or selected sensor data providers from the framework. The following code shows how to retrieve a list of all sensors available in the device:

```
#include <sensrvchannelfinder.h>
#include <sensrvchannel.h>

Library SensrvClient.lib sensrvutil.lib
```

```
CSensrvChannelFinder* SensrvChannelFinder = CSensrvChannelFinder::NewLC();
RSensrvChannelInfoList ChannelInfoList;
CleanupClosePushL( ChannelInfoList );
```


(C) Copyright Nokia 2014. All rights reserved.

```
  TSensrvChannelInfo mySearchConditions; // none, so matches all.
  SensrvChannelFinder->FindChannelsL(ChannelInfoList,mySearchConditions);

  // do something with the ChannelInfoList

  ChannelInfoList.Close();
  CleanupStack::Pop( &ChannelInfoList );
  CleanupStack::PopAndDestroy( SensrvChannelFinder );
```

To limit the sensors retrieved by the list, set the search condition by defining the criteria with the TSensrvChannelInfo argument supplied with the FindChannelsL() function. For example, to retrieve only Orientation sensors, add the following line before calling the FindChannelsL() function:

```
mySearchConditions.iChannelType = KSensrvChannelTypeIdOrientationData;
```

Another possibility for locating the required sensor is to loop the entire list and then check for an example of the type inside the loop:

```
  for ( TInt i = 0; i < ChannelInfoList.Count() ; i++ )
  {
   if(ChannelInfoList[i].iChannelType == KSensrvChannelTypeIdOrientationData)
   {
   // Do something with selected sensor.
   }
 }
```

All possible sensor types that can be found from the public S60 5th Edition SDK version 0.9 are:

- KSensrvChannelTypeIdAccelerometerDoubleTappingData
- KSensrvChannelTypeIdProximityMonitor
- KSensrvChannelTypeIdOrientationData
- KSensrvChannelTypeIdRotationData
- KSensrvChannelTypeIdMagnetometerXYZAxisData
- KSensrvChannelTypeIdMagneticNorthData
- KSensrvChannelTypeIdAmbientLightData
- KSensrvChannelTypeIdAccelerometerXYZAxisData

Utilising the Sensor Channel API

To utilise the Sensor Channel API to retrieve data from the sensor data provider, first locate the sensor data provider with the Channel Finder API. Once the sensor has been found, construct the channel to it and start listening with the following lines of code:

```
  iSensrvChannel = CSensrvChannel::NewL(ChannelInfoList[i]);
  iSensrvChannel->OpenChannelL();
```

To start listening to the channel, use the following line of code:

```
  iSensrvChannel->StartDataListeningL( this, 1,1,0);
```

To stop and close the channel, use the following two lines of code:

```
  iSensrvChannel ->StopDataListening();
```


(C) Copyright Nokia 2014. All rights reserved.

```
iSensrvChannel ->CloseChannel();
```

This code, used with the StartDataListeningL() function call, is a pointer to the MSensrvDataListener callback interface definition. This interface requires implementation of the following three functions:

```
void DataReceived( CSensrvChannel& aChannel, TInt aCount, TInt aDataLost );
void DataError( CSensrvChannel& aChannel, TSensrvErrorSeverity aError );
void GetDataListenerInterfaceL( TUid aInterfaceUid, TAny*& aInterface);
```

The GetDataListenerInterfaceL() function is for future extensions, and you can leave the implementation empty. The DataError() function is called if there are error situations within the sensor channel usage.

The S60 Sensor Framework will pass you the data from the sensor data provider in the DataReceived() function. The aChannel is a reference to the channel that is providing the data; aCount identifies the count of data that is available; and aDataLost identifies if any data from the sensor data provider is lost in an error situation.

This same callback function is called for each sensor channel that is in an active listening state, so first check the type of channel and then retrieve the data using the correct packet buffer instance. Samples of different kinds of data retrievals can be found in the example below. To retrieve orientation data, for example, implement the function like this:

```
void CExample::DataReceived( CSensrvChannel& aChannel, TInt aCount, TInt aDataLost )
{
     if ( aChannel.GetChannelInfo().iChannelType ==
KSensrvChannelTypeIdOrientationData )
    {
 TSensrvOrientationData data;
 for( TInt i = 0; i < aCount; i++ )
 {
  TPckgBuf<TSensrvOrientationData> dataBuf;
  aChannel.GetData( dataBuf );
               data = dataBuf();
  // Do something with the date in data variable
  // data.iTimeStamp
  // data.iDeviceOrientation
 }
     }
}
```

Example

Media:SensorExample 5thEd.zip

Tools

A Carbide.c++ plug-in providing sensor specific project and class templates is available for download: Media:SensorTemplates.zip

Applications

SF supported apps

Application	Author	Download link	Video demo	Device	Framework
Wellness Diary	NBL	Download	View	5th	
SF SensorEx	Nokia Wiki	Download		3rd/5th	
Archived:Mobile Extensions#Bricks	Nokia Wiki	Media:Mobile extensions preview		3rd/5th	QT for S60

(C) Copyright Nokia 2014. All rights reserved.

Extensions: Bricks demo	Nokia Wiki	extensions preview 3.zip		3rd/5th	(prerelease)
nAlertMe	Samir	Download	View	*3rd/5th	
Inclonemeter	symbianyucca	Download	View	3rd/5th	
MooLan	FH Hagenberg Michael Hölzl and Bernhard Aufreiter	Download	View	*3rd/5th	
mSense	FH Hagenberg Dominik Gusenbauer and Daniel Rothbauer	Download		5th	QT for S60
XpressBeer	kAIST	Download	View	*3rd/5th	PySymbian >v1.9.7
XpressAlarm	kAIST	Download	View	*3rd/5th	PySymbian >v1.9.7
XpressMaze	kAIST	Download		*3rd/5th	PySymbian >v1.9.7
Resco Bubbles	Resco	Download		*3rd/5th	
Marble Maze	Cahoona Games	Download		3rd/5th	
FlipSilent	Thinkchange	Download		*3rd/5th	
ShakeSMS	Thinkchange	Download		*3rd/5th	

- Doesn't run on E66 (3rd FP1)

Related articles

- Archived:S60 Platform SDK - Workarounds and Updates

(C) Copyright Nokia 2014. All rights reserved.