

Archived:GPS API in Symbian 3rd Edition



Archived: This article is [archived](#) because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template `{{ReviewForRemovalFromArchive|user=~~~~|write your reason here}}`. The article is believed to be still valid for the original topic scope.

+Before Anything+ There are many errors in this example, solution for each error is available at [Archived Talk:GPS API in Symbian 3rd Edition](#)

Overview

S60 3rd Edition has full support for [GPS](#) positioning. Both built-in and external [Bluetooth GPS](#) modules are supported. There are several [GPS](#)-enabled phones already on the market. They are Nokia 6110 and Nokia N95. More [GPS](#) phones are coming.

There is free [GPS](#) navigation software from Nokia called Smart2Go (another name is Nokia Maps). There are several 3rd party titles that also utilize [GPS](#) positioning. How can you use [GPS](#) positioning in your application?

Implementation

There are two ways. The first way is using low-level communication with external [Bluetooth GPS](#) module. It was widely used in S60 2nd Edition. It involves many technical issues and might not work correctly with some external [Bluetooth GPS](#) devices. It also will NOT work with the built-in [GPS](#) modules in Nokia 6110 and Nokia N95 phones. It's a deprecated way.

The other (recommended) way is using [Location API](#) that was introduced in S60 2nd Edition, Feature Pack 2 and also supported in S60 3rd Edition.

There are several key classes in [Location API](#). They are [RPositionServer](#), [RPositioner](#) and [TPositionInfo](#). First you need to connect to [RPositionServer](#). Then create [RPositioner](#) object and issue an asynchronous request for current location. The result will be returned in [TPositionInfo](#) structure.

Usage

Here is a high-level utility class [CGpsPositionRequest](#). It solves three problems. First, it hides all details and complexity of [Location API](#) from a user (programmer). Second, it works synchronously and removes all asynchronous complexity from the user. Third, a progress dialog (wait note) is being shown during the location request, so an end-user will see that an application is working. Here is a usage example:

```
#include "GpsPositionRequest.h"
...
// variables to hold current locations
TReal latitude, longitude;
// create CGpsPositionRequest object and put it into cleanup stack;
// pass application name as argument
CGpsPositionRequest* request = CGpsPositionRequest::NewLC(
    _L("My application"));
// get current location (this operation can be long up to 30 seconds);
// progress dialog is shown to user during this time
TBool result = request->GetCurrentPositionL(latitude, longitude);
// delete request object
CleanupStack::PopAndDestroy(request);
// process result here
if (result)
{
    // success, use latitude and longitude coordinates
}
else
{
    // failed getting current position, show error message to user
}
```

Note that you also need to include the following lines into your MMP project file:

```
SOURCE GpsPositionRequest.cpp
LIBRARY lbs.lib
```

You will also need to include [GpsPositionRequest.ra](#) file into your resource file:

```
#include "GpsPositionRequest.ra"
```

Download the source code for [CGpsPositionRequest](#) from here: [File:GpsPositionRequest.zip](#)

Another example that also has simple re-usable container that you could use for viewing the position before selecting it can be found from here: [File:GeoTagging Example.zip](#)

