



This article reprinted with permission by java.sun.com

Article

J2ME and Location-Based Services

By Qusay H. Mahmoud
March 2004

Location-based services (LBS) provide users of mobile devices personalized services tailored to their current location. They open a new market for developers, cellular network operators, and service providers to develop and deploy value-added services: advising users of current traffic conditions, supplying routing information, helping them find nearby restaurants, and many more.

This article introduces you to the field of LBS and to the Location API for J2ME (JSR 179), a set of generic APIs that can be used for developing location-based services. In addition, the article offers guidelines for designing location-based services.

What Location-Based Services Do

Location-based services answer three questions: Where am I? What's around me? How do I get there? They determine the location of the user by using one of several technologies for determining position, then use the location and other information to provide personalized applications and services. As an example, consider a wireless 911 emergency service that determines the caller's location automatically. Such a service would be extremely useful, especially to users who are far from home and don't know local landmarks. Traffic advisories, navigation help including maps and directions, and roadside assistance are natural location-based services. Other services can combine present location with information about personal preferences to help users find food, lodging, and entertainment to fit their tastes and pocketbooks.

There are two basic approaches to implementing location-based services:

1. Process location data in a server and deliver results to the device.
2. Obtain location data for a device-based application that uses it directly.

This article focuses on device-based location services.

Determining the Device's Location

To discover the location of the device, LBS must use real-time positioning methods. Accuracy depends on the method used.

Locations can be expressed in spatial terms or as text descriptions. A spatial location can be expressed in the widely used *latitude-longitude-altitude* coordinate system. Latitude is expressed as 0-90 degrees north or south of the equator, and longitude as 0-180 degrees east or west of the prime meridian, which passes through Greenwich, England. Altitude is expressed in meters above sea level. A text description is usually expressed as a street address, including city, postal code, and so on.

Applications can call on any of several types of positioning methods.

- *Using the mobile phone network:* The current cell ID can be used to identify the Base Transceiver Station (BTS) that the device is communicating with and the location of that BTS. Clearly, the accuracy of this method depends on the size of the cell, and can be quite inaccurate. A GSM cell may be anywhere from 2 to 20 kilometers in diameter. Other techniques used along with cell ID can achieve accuracy within 150 meters.
- *Using satellites:* The Global Positioning System (GPS), controlled by the US Department of Defense, uses a constellation of 24 satellites orbiting the earth. GPS determines the device's position by calculating differences in the times signals from different satellites take to reach the receiver. GPS signals are encoded, so the mobile device must be equipped with a GPS receiver. GPS is potentially the most accurate method (between 4 and 40 meters if the GPS receiver has a clear view of the sky), but it has some drawbacks: The extra hardware can be costly, consumes battery while in use, and requires some warm-up after a cold start to get an initial fix on visible satellites. It also suffers from "canyon effects" in cities, where satellite visibility is intermittent.
- *Using short-range positioning beacons:* In relatively small areas, such as a single building, a local area network can provide locations along with other services. For example, appropriately equipped devices can use Bluetooth for short-range positioning.

In addition, location methods can connect to a mobile position center that provides an interface to query for the position of the mobile subscriber. The API to the mobile position center is XML-based. While applications can be fully self-contained on the device, it's clear that a wider array of services is possible when a server-side application is part of the overall service.

Some applications don't need high accuracy, but others will be useless if the location isn't accurate enough. It's okay for the location of a tourist walking around town to be off by 30 meters, but other applications and services may demand higher accuracy.

The Location API for J2ME

The Location API for J2ME specification defines an optional package, `javax.microedition.location`, that enables developers to write wireless location-based applications and services for resource-limited devices like mobile phones, and can be implemented with any common location method. The compact and generic J2ME location APIs provide mobile applications with information about the device's present physical location and orientation (compass direction), and support the creation and use of databases of known landmarks, stored in the device.

JSR 179 requires the Connected Device Configuration (CDC) or version 1.1 of the Connected Limited Device Configuration (CLDC). CLDC 1.0 isn't adequate because it doesn't support floating-point numbers, which the API uses to represent coordinates and other measurements. The Location API doesn't depend on any particular profile -- it can be used with MIDP or the Personal Profile.

The hardware platform determines which location methods are supported. If it doesn't support at least one location provider, LBS won't be possible. Applications can request providers with particular characteristics, such as a minimum degree of accuracy. Some location methods may be free; others may entail service fees. The application should warn the user before any charges are incurred.

It is up to the application to determine the criteria for selecting the location method. Criteria fields include: accuracy, response time, need for altitude, and speed. Once the application obtains a `LocationProvider` instance that meets the criteria, it can use that object to obtain the location, in either of two ways:

- Invoke a method synchronously to get a single location.
- Register a listener and get periodic updates at application-defined intervals.

The `Location` class abstracts the location results. Its object contains coordinates, speed if available, textual address if available, and a time stamp that indicates when the location measurements were made.

Coordinates are represented by either of two classes:

- A `Coordinates` object represents a point's latitude and longitude in degrees, and altitude in meters.
- A `QualifiedCoordinates` object contains latitude, longitude, and altitude, and also an indication of

their accuracy, represented as the radius of an area.

The following segment of code demonstrates how to obtain the present location of the device synchronously:

```
...

// Set criteria for selecting a location provider:
// accurate to 500 meters horizontally
Criteria cr= new Criteria();
cr.setHorizontalAccuracy(500);

// Get an instance of the provider
LocationProvider lp= LocationProvider.getInstance(cr);

// Request the location, setting a one-minute timeout
Location l = lp.getLocation(60);
Coordinates c = l.getQualifiedCoordinates();

if(c != null ) {
    // Use coordinate information
    double lat = c.getLatitude();
    double lon = c.getLongitude();
}
...
```

Landmarks

A *landmark* is a location associated with a name and a description. Landmarks can be stored in a device-based database, where they can be shared among all J2ME applications. Landmarks can store frequently used locations: home, office, favorite restaurants, and so on. Each is represented by a `Landmark` instance, and the database by a `LandmarkStore`. You can create multiple named `LandmarkStores` to group locations into categories such as cinemas, museums, or customer sites.

If the device includes a compass, the application may be able to determine not only its location but its orientation, which is useful in navigational applications. The `Orientation` class represents the device's azimuth as an angle from due north, which the application can easily convert to a compass direction.

Security and Privacy

Many users consider location information to be highly sensitive, and are concerned about a number of privacy issues, including:

- *Target marketing:* Mobile users' locations can be used to classify customers for focused marketing efforts.
- *Embarrassment:* One customer's knowledge of another's location may lead to embarrassing situations.
- *Harassment:* Location information can be used to harass or attack a user.
- *Service denial:* A health insurance firm might deny a claim if it learned that a user visited a high-risk area.
- *Legal restrictions:* Some countries regulate the use of personal data.

For these and other reasons, users must know when their location is given to an application.

Guidelines

Keep the following guidelines in mind when designing location-based services:

- Handle unavailability of services gracefully. The user's location may not always be available, for any of several reasons;
 - The device is cut off from any of the location methods it supports, in a tunnel or on an airplane for example.
 - The user withholds permission to release the information.
 - No location provider that the device supports is available.
- Depending on the method used, determining the location may take a long time. The delay may be so long that the end result isn't useful in, for example, a navigation application. Keep the user informed.
- Location service fees, typical of network-assisted location methods, can add up quickly, so don't overuse fee-based services.
- Be sensitive to privacy concerns.
 - Tell customers about the information being collected on them and how it will be used.
 - Offer customers the choice of what location information to disclose, and when appropriate an option not to participate.
 - Allow customers to review their permission profiles so that they know what they are permitting.
 - Protect location information so that it cannot be accessed by unauthorized persons.

You should also take full advantage of the MIDP 2.0 security framework, which restricts the application's access to location data to cases in which the user explicitly confirms permission.

Summary

Through the Location API for J2ME, you can use information about the user's position to build new kinds of applications and services for mobile devices such as cell phones and PDAs, and to enhance existing services. JSR 179 specifies a generic API for obtaining locations, and thus makes porting LBS applications to a wide range of devices much easier. The critical issue that LBS developers must address is the privacy of the customer. To ensure privacy, follow sound programming guidelines and use the security framework in MIDP 2.0. For More Information

- [J2ME](#)
- [Mobile Information Device Profile 2.0 \(JSR 118\)](#)
- [Location API for J2ME specification \(JSR 179\)](#)

Acknowledgments

Special thanks to Gary Adams of Sun Microsystems, whose feedback helped me improve this article.

About the Author: [Qusay H. Mahmoud](#) provides Java consulting and training services. He has published dozens of articles on Java, and is the author of *Distributed Programming with Java* (Manning Publications, 1999) and *Learning Wireless Java* (O'Reilly, 2002).