



Voyager Geolocation Developer's Guide

Version 1.0 for Voyager 8.0-RC3

Table of Contents

Introduction	4
Overview.....	4
Preface.....	4
Geolocation Requirements and Support	4
Java	4
.NET.....	4
Contacting Technical Support	4
Voyager Geolocation API Overview	5
Key interfaces and Classes.....	5
Using Geolocation APIs	5
Appendices – Examples.....	7
LocationExample1/Location1Example.....	7
LocationExample2/Location2Example.....	7
LocationExample3/Location3Example.....	7
Source code location.....	8

<This page intentionally left blank>

Introduction

Overview

The Voyager™ Geolocation API provides a Java or .NET agent or application the ability to obtain Geolocation information from local or remote geolocation-aware devices.

Preface

The purpose of this manual is to provide an introduction to the Voyager Geolocation API. This guide assumes basic knowledge of geolocation device capabilities and functions.

This preface covers the following topics:

- Voyager Geolocation Requirements and Support
- Contacting technical support

Geolocation Requirements and Support

Before attempting to use the geolocation features of Voyager:

Java

- A Java Development Kit (JDK) installed on your computer. VOYAGER requires JDK 1.4.2 or later. You can download the latest release of the JDK from www.javasoft.com at no charge.
- A working version of Voyager installed.
- At least one location-aware device supported by Voyager.

.NET

- .NET framework 2.0
- A working version of Voyager installed.
- At least one location-aware device supported by Voyager.

Contacting Technical Support

Copyright © 2007 – 2010 Recursion Software, Inc.
All Rights Reserved

Recursion Software welcomes your problem reports, and appreciates all comments and suggestions for improving VOYAGER. Please send all feedback to the Recursion Software Technical Support department.

Technical support for VOYAGER is available via the web, email, and phone. You can contact Technical Support by sending email to psupport@recursionsw.com or by calling (972) 731-8800.

Voyager Geolocation API Overview

The Voyager Geolocation API provides a consistent API around native location provider API's including JSR 179 for the Java CLDC environment and the GPS Intermediate Driver API for the .NET environment. The Voyager Geolocation API supports local and remote access to a device's location.

The goals of the Geolocation API are:

- Simplify access to location information on supported platforms/devices.
- Provide a consistent API to access and manipulate location information.
- Support access to location information from remote devices.

Key interfaces and Classes

- **LocationService** – The `LocationService` class is used to obtain the `ILocationProviderManager` for the location environment.
- **ILocationProviderManager** – Manages implementations of the `ILocationProvider` interfaces and provides access to them.
- **ILocationProvider** – Provides location information through direct query and listener.
- **ILocationListener** – Implement this to be notified of updates to a device's location.
- **Location** – Provides information about a given geolocation reading or fix.

An `ILocationProvider` implementation provides access to location information. There may be multiple implementations of this interface, or multiple instances of a single implementation, based on a platform's native location capabilities. The `ILocationProviderManager` implementation manages these various instances and is used to obtain a specific `ILocationProvider`. The `ILocationProviderManager` in turn is managed by the `LocationService`, a Voyager service that is started when `Voyager.startup()` is called and stopped when `Voyager.shutdown()` is called.

Using Geolocation APIs

Copyright © 2007 – 2010 Recursion Software, Inc.
All Rights Reserved

The `LocationService` and `ILocationProviderManager` are used to obtain an `ILocationProvider`. To acquire an `ILocationProvider` perform the following steps:

1. Obtain the singleton `LocationService` instance.
2. Obtain the `ILocationProviderManager` managed by the `LocationService`.
3. Acquire an `ILocationProvider` from the manager.

In code, this takes the following form:

Java/C#

```
LocationService service = LocationService.getInstance();
ILocationProviderManager manager = service.getProviderManager();
ILocationProvider provider = manager.acquireProvider();
```

Note that there are several variants of the `acquireProvider()` method:

`acquireProvider()` – Acquire the default local location provider

`acquireProvider(Object o)` – Acquire the location provider local to the given object. If the object is local to the process, this returns the default local location provider, creating it if it does not already exist. If the object is a proxy, this returns the location provider local to (in the same process as) the proxy's referent object, creating it if it does not already exist.

`acquireProvider(Object o, String providerName)` – Acquire the location provider local to the given object with the given provider name. If the object is local to the process, this returns the named location provider, creating it if it does not already exist. If the object is a proxy, this returns the named location provider local to (in the same process as) the proxy's referent object, creating it if it does not exist.

`acquireProviderAt(String url)` – Acquire the default location provider at the given server url, creating it if it does not exist.

`acquireProviderAt(String url, String providerName)` – Acquire the named location provider at the given server url, creating it if it does not exist.

An `ILocationProvider` is identified by a unique provider name. For convenience, three standard provider names are provided as constants in `LocationConstants`: `defaultProvider`, `bestFixProvider`, and `lowestCostProvider`. You can register/deregister providers (or override the providers registered under the standard provider names) with the `AbstractLocationProviderManager.registerProvider()` method. (Unless you provide a separate implementation, the `ILocationProviderManager` returned from `LocationService.getProviderManager()` will be an `AbstractLocationProviderManager`.)

Copyright © 2007 – 2010 Recursion Software, Inc.
All Rights Reserved

Once you have acquired an `ILocationProvider` you can obtain geolocation information (if it is available) via several methods:

`getCurrentLocation()` – Query the geolocation provider for the current location. This method typically invokes the ‘native’ location services on the device and should be used with care.

`getLastKnownLocation()` – Query the geolocation provider for the location last obtained via a call to `getCurrentLocation()` or via a listener update (see below). This method returns a cached value and can be called without invoking the ‘native’ location services on the device.

`setLocationListener()` – Set a location listener that will be updated regularly. This method typically results in the ‘native’ location service being invoked on a periodic basis and should be used with care. Note that the periodicity requested may not be adhered to depending on device timeouts and/or thread scheduling. Each provider may have at most a single listener. Utilize care when setting listeners to multiple providers as they may utilize the same underlying ‘native’ location services.

The `Location` class contains all geolocation information from a given reading/fix. Most modern devices use GPS satellites to acquire a fix and provide accurate and reliable readings. In this case the `Location` will have a valid `GPSInformation` field containing additional details about the fix obtained. The `Location` may also provide address information via an `AddressInformation` instance.

Appendices – Examples

This appendix provides an overview of the Geolocation API examples.

LocationExample1/Location1Example

This example demonstrates obtaining an `ILocationProviderManager` and local `ILocationProvider`, as well as obtaining a `Location` from an `ILocationProvider`.

LocationExample2/Location2Example

This example demonstrates obtaining a remote `ILocationProvider` and a `Location` from that provider.

LocationExample3/Location3Example

This example demonstrates setting (and removing) a listener on a local `ILocationProvider`.

Source code location

After you install Voyager, the source code for the examples can be found in the following locations under your Voyager installation:

Java (JSE)

```
examples/java/se-cdc/java/examples/geolocation/LocationExample1.java
examples/java/se-cdc/java/examples/geolocation/LocationExample2.java
examples/java/se-cdc/java/examples/geolocation/LocationExample3.java
```

Java (CLDC)

```
examples/java/cldc/java/examples/geolocation/LocationExample1.java
examples/java/cldc/java/examples/geolocation/LocationExample2.java
examples/java/cldc/java/examples/geolocation/LocationExample3.java
```

C# (CF)

```
examples/cf/csharp/Common/examples/Location1Example.cs
examples/cf/csharp/Common/examples/Location2Example.cs
examples/cf/csharp/Common/examples/Location3Example.cs
examples/cf/csharp/Common/LocationListener.cs
```