RECURSION
SOFTWARE, Inc.

Universal Services & The Connected World:

Machine 2 Machine Knowledge Networks, Dynamic Discovery and Ad-Hoc Data Sharing

Mike Eddings
Director, Government & Solutions

Executive Summary

Continuous performance advancements in networking and computing, combined with market forces driving decreasing costs and accelerated adoption of mobile and embedded processing capabilities have created a perfect opportunity for breakthrough applications involving automatic communications between devices and advanced decision making facilitated by Machine 2 Machine Knowledge Networks (M2MKN).

Advanced mobile and embedded devices coupled with agent-based intelligent middleware can enable collaborative device communities that discover, communicate and collaborate in real-time and push relevant content from peer-to-peer and peer-to-group regardless of platform or operating system on the machines or devices.

Contents

Insights and Opportunities

The next-generation of software development will see applications that combine the best of data gathering, location-based services, universal business objects, dynamic discovery, and other forward looking capabilities that intelligent middleware is beginning to address. Leveraging such capabilities across applications and devices will be a huge step forward from existing applications that mashup web search, GPS positioning, mapping software, and SMS messaging, without truly being dynamic or ubiquitous in nature. Today's mobile and embedded applications are dependent upon the Internet so they are constrained by intermittent network access, reliance on static yellow page directories and mapping databases. Ultimately, they suffer from the inability to provide ad-hoc device discovery.

Any-to-Any Data Sharing

Enter peer-to-peer distributed ad-hoc networks. While the Internet is on a different time horizon, one could argue that a reactive timeframe, distributed ad-hoc networks would bring any flavor of mobile apps into the "now". Such collaborative device communities provide real-time location-based device discovery in which a device is actively entering, leaving and seeking other devices within a community.

The next evolution of location-based services will allow "edge" devices, either business or consumer, human or machine operated, wireless or embedded, to participate in coordinated knowledge sharing, problem solving and transactions that span dynamically assembled communities of devices and multiple companies/ organizations and/or consumers.

A software platform that is device and language agnostic is necessary to unify and transform the wide range of smart devices into wireless learning machines that form intelligent device communities. A platform must also enable server-level intelligence on small footprint devices that can learn data patterns or user behaviors and habits. Based on patterns and history, profiles, preferences and the state (location, work, play, time or day, etc.) these devices can deduce important events and needs and proactively deliver content.

Machine 2 Machine Knowledge Networks (M2MKN)

Most embedded/M2M applications run on a mix of hardware that typically includes embedded systems devices, like, power and water meters or security and traffic cameras, and, increasingly, mobile laptop PCs and smart phones, communicating over wireless connections with centralized or distributed server computers with attached workstation clients.

Without edge computing middleware, solutions are usually built as client/server systems in which all the end-point devices communicate with the server(s), either through tightly-coupled, early-bound protocols like remote procedure calls or late-bound, loosely-coupled protocols like web services. The end-point devices do not typically communicate directly with each other. To the limited extent that they inter-operate at all, they do so through, and under the control of a server.

- Data collection devices like meters, even though they may have considerable communications and computing capabilities, often just capture values to a buffer and periodically they either connect to the server or are polled by the server to transmit collected data, doing little else on their own.

- In some cases, a device like a meter might have the intelligence to recognize an important event, like an excessively high value that might indicate a water leak or power theft, and to generate an alarm notification. In a client/server configuration, the meter would notify the server and the server would in turn initiate corrective action.
- Besides meters or other embedded system devices, an M2M system might incorporate other edge devices like mobile phones, generic laptops or specialized hand-held computers.
- In exceptional cases like an alarm response, or routine cases like scheduled service, the server might communicate directly with the mobile devices to direct field personnel, or it might instead communicate with a co-located wired workstation to notify a dispatcher who then directs the field personnel by telephone calls or push-to-talk communications.

The term "machine-to-machine" suggests a level of automation that is seldom actually achieved in so-designated systems and applications. In most cases, intermediate human actions and person-to-person voice communication are common, indispensable elements of the business processes that these solutions ostensibly automate. In practice, they might more accurately be described as machine-to-person-to-machine systems. This is due in some part to the nature of the client/server networked computing model on which they have been based.

- It dictates conceptually unnecessary indirection of commands and communications that is difficult and costly to automate.
- The meter notified the server, the server alerts the dispatcher through his workstation, the dispatcher uses a push-to-talk telephone to broadcast a notice to all workers, available workers communicate their locations to the dispatcher, the dispatcher identifies and alerts the worker closest to the meter and submits a work order to the server through his workstation, the server processes the work order and reassigns or reschedules the diverted worker's pending task and notifies the dispatcher of the new assignment.

With intelligent middleware, this scenario is considerably simpler:

- The meter finds and alerts the closest worker through his smart phone and notifies the server system of the problem, the diverted worker acknowledges the new assignment to the server, the server produces an appropriate work order, reassigns the diverted worker's pending task and notifies assignee through his smart phone and sends the corresponding work order to his laptop.
- The ability to abstract or enable multi-platform location services, peer-to-peer and peer-to-group communications, and unified cross-device applications programming interface combine to make this an easier, less expensive application to build, deploy, support and modify.
- Equally important, it eliminates a substantial dependence on manual steps and intermediate human action, yielding a considerable reduction in the operations cost to business.

Potential Applications

1) **Multi-National, Machine-to-Machine Knowledge Grid:** Providing data processing and secure information sharing across a large, heterogeneous network of databases, servers, mobile devices and smart sensors.

   Key requirements: Non-intrusive integration, single port firewall navigation, multi-protocol support and forward deployment of lightweight rule sets and processing for automated data retrieval and distribution.

2) **Field Service Automation & Collaboration:** Providing guaranteed alarm delivery, acknowledgement and automated group collaboration to field technicians and engineers.

   Key requirements: Location awareness, dynamic discovery, support for embedded operating systems and multiple wireless devices with guaranteed messaging to linked and nested user groups.

3) **Wireless, Automatic File Sharing Across Family/SMB Devices:** Providing a differentiator to a storage media company desiring an application that allows for automatic, wireless sharing of files from/to SD cards on personal/family devices and the Home/SMB computer.

Key requirements: Cross platform support across all devices ability to share content to/from personal/family/SMB devices w/out user intervention.

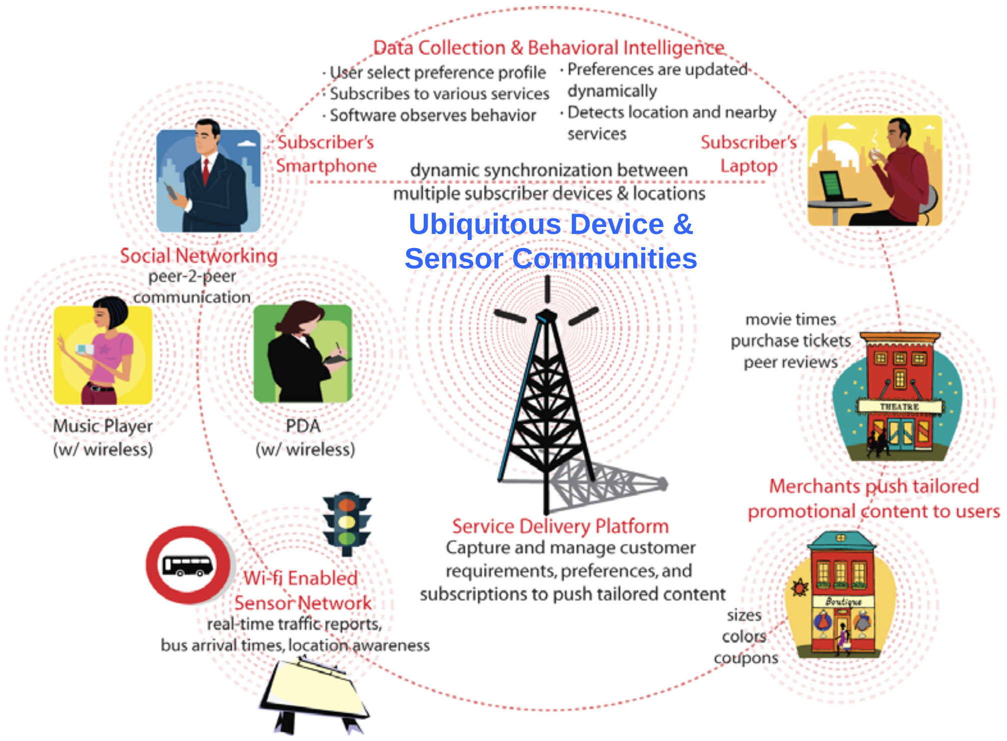
The Perfect Storm

In light of the advancements in high speed networks and processing capabilities on embedded, a perfect storm is brewing that will enable the next-generation of services. The missing piece is a software platform that can act as an intelligent software abstraction and communications layer for all wireless networks (not just telco), as well as wireless and embedded devices. A platform must enable any wireless device, including Smartphones, MIDs, UMPCs, laptops, set-top boxes, gaming devices, embedded devices, sensors, RFID readers, etc., to share information seamlessly peer-to-peer in real-time.

Imagine a platform that can run applications and provide and/or display relevant, timely, location sensitive content to you at any time, wherever you are, and with whatever device you are using or viewing. The device could be the laptop, wireless phone, TV, Smartcar, or a sign you pass in the mall or on the road.

This advanced technology combined with an open access network would provide a super highway for apps, knowledge sharing, and social and business transactions. It would connect people person-to-person, person-to-business, or business-to-business. It can turn everyone into an advertiser and social networker and discover other entities with compatible interests, careers, services, etc.

A company with the right vision and the right platform will be unstoppable in delivering new applications and intelligent, content delivery and knowledge sharing to people wherever they are, with whatever device they have and whatever wired/wireless networks it supports.


Data Collection & Behavioral Intelligence
· User select preference profile
· Subscribes to various services
· Software observes behavior
· Preferences are updated dynamically
· Detects location and nearby services
Subscriber's Smartphone
Subscriber's Laptop
dynamic synchronization between multiple subscriber devices & locations
Ubiquitous Device & Sensor Communities
Social Networking
peer-2-peer communication
Music Player (w/ wireless)
PDA (w/ wireless)
movie times
purchase tickets
peer reviews
THEATRE
Merchants push tailored promotional content to users
Service Delivery Platform
Capture and manage customer requirements, preferences, and subscriptions to push tailored content
Wi-fi Enabled Sensor Network
real-time traffic reports, bus arrival times, location awareness
sizes
colors
coupons
Boutique

Intelligent M2M Platform Requirements: A Top 10 List

While today's M2M applications are limited by their technology, a next-generation software platform, which can abstract wireless networks, operating systems and devices and provide intelligent messaging will enable the next wave of context-aware applications and advertising. The following is a checklist for choosing a next-generation platform that can handle the demands of a networked world.

1. Wireless/Embedded Device Support

A true peer-to-peer application must be able to run, in some form, on all devices. To do so, its software platform must be pervasive and supported on these same devices. Software agents comprising these applications, also need to be able to run in the popular embedded software stacks, such as Java's Micro Edition, Microsoft's Compact Framework, and OSGI Containers, on a wide range of embedded operating systems such as Windows Mobile, Symbian, Android, LIMO, Embedded Linux versions, and Sun's JavaFX to name just a few.

2. Decentralized and Centralized Messaging w/Ad-Hoc Communities

A next-generation platform must enable communication between groups of devices/systems without the need of a centralized messaging server. It must support the creation of ad-hoc communities of devices or nodes, as well as the ability to support filtering of messages across these communities.

The platform must also support passing messages over standard centralized messaging servers for integration with enterprise and legacy systems. More specifically, there needs to be seamless integration with Microsoft's Message Queue (MSMQ), Java's Message Server (JMS) and Object Management Group (OMG)'s Data Description Service (DDS), etc. Support must exist for devices joining and leaving the network, which will result in changing "internet" addresses. Integration with a SIP Server is the way to provide dynamic IP support but voice, data, and chat support over IP.

3. Mobile SOA Architecture

A next-generation platform must provide a Service-Oriented Architecture (SOA). These services need to be accessible via centralized Web Service Container such as Microsoft's Internet Information Services (IIS) and those provided in the Java world, but also must be accessible in a decentralized fashion directly to agents, exposed as web services, that are running on edge and wireless devices. The location of intelligent mobile agents and the Mobile Web Services they expose must be irrelevant to the web service client. Finally, all agents need to be accessible by a Service Description in a Yellow-pages directory, ideally one that is Universal Description, Discovery, and Integration (UDDI)- compliant.

4. Increase Network Survivability and Mobility

A next-generation platform must enable data processing at the source to minimize network traffic, handle unreliable and/or limited network connections, and adjust to hardware failures or CPU load. Therefore, these devices must be able to persist data via a micro relational or objects database. Additionally, the software components or agents running on edge devices need to support multiple wireless protocols (GSM, CDMA, Wi-Fi, UWB, Bluetooth, NFC, RFID, etc.) and associated networks (telco, Wide Area, Local, Personal, etc.). Ideally, they will dynamically reconfigure themselves to use a communication protocol that best matches the capabilities of their current network connection and the current node(s) they are in communication with.

5. Security

In a ubiquitously networked world, it will often be necessary to maintain data on edge devices. The security concerns facing enterprises today will need to incorporate solutions that extend into a collaborative environment. Applications must provide an extremely high level of security to ensure privacy and protection from rogue/viral clients and software agents. This will involve security agents and agent managers that provide capabilities above and beyond the current encryption, authentication, and authorization that are currently employed in today's centralized client server applications.

6. Provide a Single Unified Platform for .NET, Java and C++

These sophisticated applications cannot be limited to a single development environment and programming language. Furthermore, the same API should be provided to .NET, Java and C++

developers. The platform should support the same set of collections and algorithms across these languages as well. This would greatly increase programmer productivity and allow developers from all camps to easily work together and share software. The platform must seamlessly integrate with .NET, JEE and legacy (MVS, CORBA, etc.) enterprise systems and services, and any combination thereof, in either a traditional Web Services architecture, or in a high performance manner using binary protocols. These Mobile 2.0 applications may often need to communicate with more than one enterprise or organization.

7. Location Engine

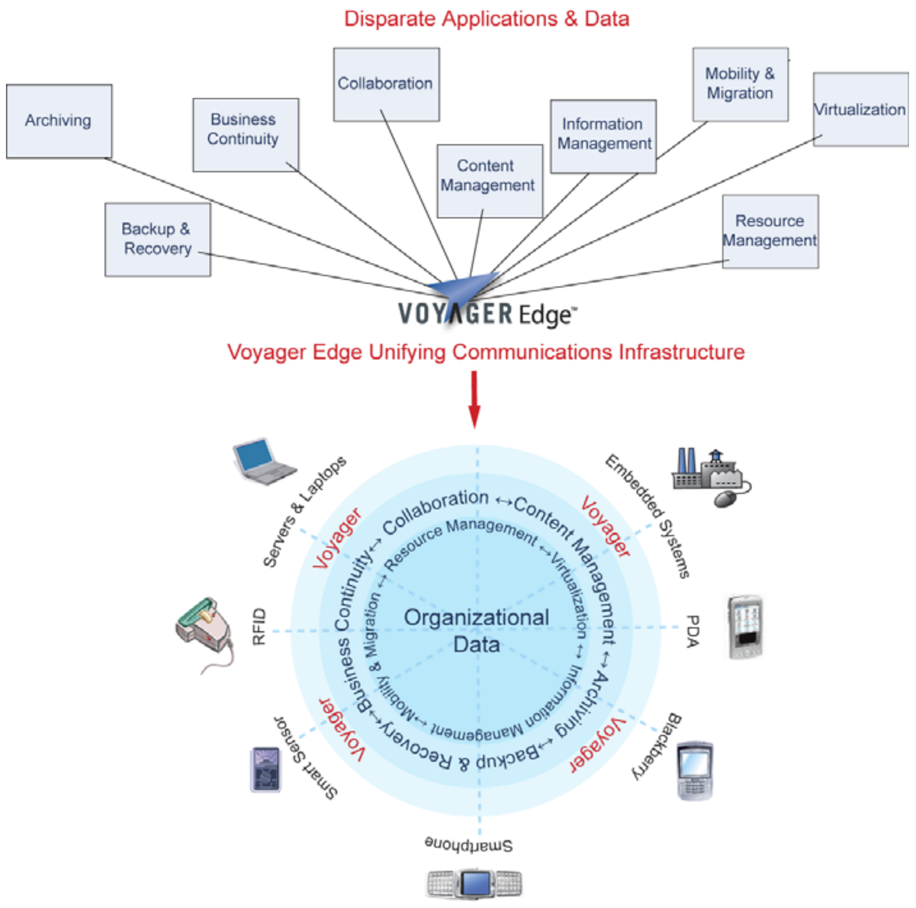
To enable location-aware, community-based applications, the pervasive platform should provide a location engine and service that allows determination of any node's location from any other node. It should be a simple API that is identical across all platforms, languages and devices regardless of how a particular node determines where it is.

8. Transactional at the Edge

A next-generation platform must extend transactions from the enterprise to include edge devices allowing for distributed, but coordinated tasks among peers, peer groups, and the enterprise. Support must be provided to allow for intelligent agents running on the edge to participate in guaranteed message delivery with XA-compliant enterprise transaction managers such as those provided in .NET's Microsoft Transaction Service (MTS), Java's Transaction Service (JTS), and OMG's Object Transaction Service (OTS).

9. Artificial/Cognitive Intelligence

A next-generation application will need to utilize intelligent software agents that can gather data, respond quickly based on this data as it changes, produce and distribute knowledge, and possibly initiate other agent activities. The underlying rules engine must be easy to use, provide very high performance against potentially large rule sets, and must be available in multiple languages. These agents should also be able to understand different ontologies as a powerful abstraction to the modeling of knowledge about different domains.



10. Embedded and Enterprise Database Integration and Synchronization

A next-generation platform must provide a simple way to access databases, regardless of the type of database whether it is a relational, object, XML or a multi-user enterprise database or single-user embedded. Developers need to be shielded from the intricacies that exist with these various flavors and have support for data synchronization between the edge and enterprise.

Convergence of Devices, Platforms, and Messaging

An intelligent, distributed computing platform that can provide these capabilities, will provide a superhighway for software engineers to create the next-generation of applications and deliver intelligent, location-based information to any person, anywhere, regardless of the device they are using.

Long considered a technology before its time, Voyager (formerly Voyager ORB) from Recursion Software is an agent-based software platform that has been used to build intelligent applications since late-1990s. The latest version, Voyager, extends server-level intelligence to a wide range of device communities, embedded and wireless operating systems, and software languages using varied distributed protocols and messaging capabilities.

Software architects and engineers using Voyager have the flexibility to freely develop dynamic, intelligent, and decentralized applications in both .NET and Java, on the devices and servers they need to target. This is possible whether the device/node is a desktop, PDA, Smartphones, MIDs/UMPCs, set-top box, RFID reader, mall or road sign, embedded sensor or smart-tag (see *Getting Started with Voyager* for code sample).

By utilizing Voyager, developers will have at their disposal a choice of wireless networks, distributed protocols, and decentralized and centralized messaging capabilities to leverage in creating the ad-hoc, distributed, social, knowledge, and problem-solving networks of tomorrow. Combined with rules engines, such as a RETE-based rules engine that is integrated within the Voyager platform and location engine, engineers can also weave artificial intelligence and cognitive capabilities into the location-based software they deploy to all targeted devices and servers. Engineers will be able to produce applications that provide real-time intelligence, situational awareness, and coordination at the edge not found today.

Legacy Integration and Mobilization

Enterprise software stacks, like software languages, continue to be born, but old ones never go away. JEE and .NET architectures and related languages dominate, yet CORBA and Mainframe systems and their associated languages are still prevalent. A very similar situation exists in the device world where Windows Mobile, Symbian, Embedded Linux, Blackberry, iPhone, Android, and most recently Sun's JavaFX, compete with many other embedded operating systems. The same is true for wireless networks like Bluetooth, Wi-Fi, and the emerging Near Field Communication (NFC) and WiMax networks.

Using Voyager as a communications backbone will provide a single cohesive and comprehensive runtime environment that allows next- generation applications to seamlessly integrate in this diverse software and device environment, and the increasingly varied wired and wireless networks used to connect them.

A World of Intelligent Location-Based Applications

The potential impact of ubiquitous computing reaches beyond mobile services and into several industries such as telecom management, healthcare, energy, transportation, insurance, education, finance and entertainment industries. Some industry insiders predict that intelligent location-based applications remain a goal to be obtained over the next few years, yet the architecture for such next-generation applications exists today for those willing to take the first step toward this world.▪

Getting Started with Voyager: Sample Code

Believing that Voyager can enable these next-generation intelligent apps is best demonstrated by a real-world example. To get started, download a free community edition of Voyager. It comes with extensive documentation and over 40 examples covering all of the features of Voyager, including examples in Java, C#, VB.NET and Managed C++.

1. Creating, naming and moving a mobile object on/to any node

The following is a code snippet to:

1. create an object
2. bind it to our federated, decentralized Naming Service
3. move it

```
// create a remote component on the
local node
IStockmarket market1 =
(IStockmarket)
Factory.create("examples.stockmarket
.Stockmarket");

// create an remote component on
node "//dallas:8000"
IStockmarket market2 =
(IStockmarket)
Factory.create("examples.stockmarket
.Stockmarket", "//dallas:8000");

// bind to naming service
Namespace.bind("//dallas:8000/NASDAQ
", market );

// move to another node "//
tokyo:9000"
IMobility mobility = Mobility.of(
market ); // obtain mobility facet
3b.mobility.moveTo("//tokyo:9000" );
// move the object to a new location
```

2. Turning a remote component into an autonomous agent

The following is a code snippet to:

1. create an object
2. bind it to our federated, decentralized Naming Service
3. turns it into an agent
4. makes it autonomous
5. move it to another node

```
// create a remote trader component
on the local node
ITrader trader = (ITrader)
Factory.create(
Trader.class.getName() );

// create an Agent facet
IAgent aAgent =
AgentFacet.of(trader);

// setting to autonomous, prevents
it from being garbage collected
// default is true
aAgent.setAutonomous(true);

// move to node and reset it at
arrival by calling "atMarket"
method
aAgent.moveTo("//dallas:8000",
"atMarket" );
```

3. Creating and joining an object/agent community (a.k.a space) on any node

The following is a code snippet to:

1. create an agent space on a remote node
2. create a remote component and add it that that community
3. create another agent space on a remote node
4. create another remote component and add it that the community just created

```
// creating a space on a remote node
ISubspace subspace1 = (ISubspace)
Factory.create(
"com.recursionsw.ve.space.Subspace"
, "8000/Subspace1" );

// creating a remote component on
that node
IConsumer consumer1 = (IConsumer)
Factory.create(
"examples.space.Consumer", new
Object[]{ "jack" }, "8000/Jack" );

// adding the remote object to that
remote space
subspace1.add( consumer1 );

// creating another space on another
remote node
ISubspace subspace2 = (ISubspace)
Factory.create(
"com.recursionsw.ve.space.Subspace"
, "8000/Subspace1" );

// creating a remote component on
that node
IConsumer consumer2 = (IConsumer)
Factory.create(
"examples.space.Consumer", new
Object[]{ "jack" }, "8000/Jack" );

// adding the remote object to that
remote space
subspace2.add( consumer2 );
```

4. Joining two object/agent communities on any nodes

The following is a code snippet to:

1. connect or chain the two spaces that were previously created on different nodes

```
// connecting two agent communities
together
// regardless of what nodes they
are located on
subspace1.connect( subspace2 );
```

5. Multicasting a message within an object/agent community

The following is a code snippet to:

1. lookup an object/agent community
2. get a multicast proxy
3. call a method on any objects of type “consumer” located in that community

```
// looking up an agent community
ISubspace subspace1 = (ISubspace)
Namespace.lookup( "8000/Subspace1"
);

// get the multicast proxy
associated with Consumer agent
IConsumer consumer1 = (IConsumer)
subspace1.getMulticastProxy(

      "examples.space.IConsumer" );

// publish the message to the
community
// calls “news” method on every
Consumer object in the community
Multicast.invoke( subspace1,
"news", new Object[] { "newsflash
2!" },

      "examples.space.IConsumer" );
```

6. Exposing and accessing an object/agent via its automatically created Web Service

The following is a code snippet to:

1. enable web services support
2. call a method on a remote component using SOAP

```
// enable web services support
com.recursionsw.ve.web.services.Web
Services.enableWebServices();

// Create a service model for the
web service client
ServiceFactory serviceFactory = new
ObjectServiceFactory();
Service serviceModel =
serviceFactory.create(IStockmarket.c
lass);

// Create a client proxy
XFireProxyFactory proxyFactory = new
XFireProxyFactory();

String hostName =
InetAddress.getLocalHost().getHostNa
me();
IStockmarket market =
      (IStockmarket)proxyFactory.cre
ate(serviceModel,
      "http://dallas.recursionsw.com
:8000/services/Stockmarket");

// call “getQuote” using on
IstockMarket Web Service using SOAP
market.getQuote("sun");
```

RECURSION SOFTWARE, Inc.

About Recursion Software, Inc.

Recursion Software is an innovative provider of intelligent middleware and distributed computing solutions based on Service Oriented Architecture (SOA) principles and interoperability standards. Since 1993, our products have enabled enterprises to extend their current application architecture while providing the tools developers need to build the next-generation of intelligent, mobile applications. The company is a small, privately held corporation, located in the Dallas-Fort Worth area.

Recursion Software is regarded for its Voyager platform, a powerful agent-based interoperable platform that supports a total range of edge devices, including handheld devices, PDAs, sensors, cameras, and other wireless devices. The company remains the leading proponent and preferred platform for intelligent mobile agent and agent community technology and has 47 patents issued and/or pending.

For more information, visit our website at recursionsw.com