Beyond the Internet: Building Multiplatform P2P Communities

Bob DeAnna, Chief Technology Officer
Victoria Romero-Gomez, Marketing Manager

Gartner predicts a future of mobile 'ambient business', where consumers explore their environment to find relevant value propositions...Data will be selectively pushed to the user based on context, matching the users' needs, interests, mood, location and even recent behavior...Location, combined with personal presence, creates transient communities, which offers an opportunity for better use of [wireless] time or marketing.

-- Nick Jones, Gartner Says Corporate Mobility Becomes Mainstream and Outlines the Shape of the Future with Mobile Business 2.0, April 3, 2006

Executive Summary
Autonomous intelligent devices communicating, collaborating and discovering each other is a world vision hitherto relegated to the Sci-Fi channel. Yet, with recent breakthroughs in intelligent middleware, device communities can extend beyond the Internet by communicating peer-to-peer regardless of platform or OS. Agent-based intelligent middleware can enable collaborative device communities that communicate in real-time and push content without requiring the user to 'Google' for, or even think about it. Read this white paper to learn the basic technology requirements to enable intelligent apps and prepare your organization for the next generation of community-based software development.
Insights and Opportunities
The next-generation of software development will see applications that combine the best of social networking, 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 communities 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 apps 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. This realization begs the question: how good are apps like mobile social networking (MoSoSo) if your device can’t find new friends? The true power lies in the ability for a wireless device to autonomously discover another wireless device in your immediate location in real-time.

Even Google, the undisputed leader in mobile search, has yet to incorporate true context awareness. Today’s mobile search and advertising is not much different from the desktop model that is passive and historical, not proactive and immediate. It forces you to know your location and enter your query then pulls previously published content from a database. GPS location technology alone only solves one part of the equation. In most cases, search results offer too much information. After all, what good are 600,000 page hits if you’re only interested in a few highly relevant search results? For some applications, like email and instant messaging, these limitations are perfectly acceptable. However, for location-based, time-sensitive applications, like MoSoSo or mobile search/advertising, the moment of opportunity or the person or business of greatest value could quickly pass you by. Location-based filtering, personalized search and content delivery, and integration with inventory systems will filter out searches by context, preference, and availability.

Peer-to-Peer is Here!
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 multiple mobile devices of professionals and consumers into wireless learning machines that form intelligent device communities. A platform must also enable server-level intelligence on small footprint devices that can learn the user’s behaviors and habits. Based on a user’s history, their opt-in preference profile, and the user’s state (location, work, play, time or day, etc.) these devices can deduce the user’s needs and proactively deliver content without requiring the user to ‘Google’ for, or even think about.

Will companies like Google cease to be relevant in the world of push content? Quite the contrary. Google or LBS pioneers might literally end up powering it, not only threaten Microsoft in the process, but also telco carrier giants such as AT&T and Verizon (see Google: The Next Microsoft and AT&T?)
Google: The Next Microsoft and AT&T?

Although Google’s core competency is advertising, it has quickly become a leader in web and mobile app development. For the moment, these apps are just another means for attracting more views to the advertisements they host or deliver. Devices, OSs, wireless networks, web sites and even applications are simply the means to deliver their advertisements.

Why did Google take a much-publicized open access position on the Federal Communications Commission (FCC) auction of wireless spectrum? The federal government’s upcoming auction of wireless spectrum in the 700-megahertz (MHz) band is part of the transition from analog to digital TV broadcasts that must be complete by February 2009. This band is arguably the last chance for new entrants in the cell phone market to gain equal access. From their public policy blog, Google suggested that the following four open standards should apply to the license conditions:

1. **Open applications**: There should be no restrictions to consumers on what software applications, content, or services that can download and use.
2. **Open devices**: Consumers should be free to use any wireless network they prefer in the 700 MHZ range on their handheld communications device;
3. **Open services**: Resellers should be able to acquire wireless services from a 700 MHz licensee at wholesale prices.
4. **Open networks**: Internet service providers and other third parties should be able to tap in at any technically feasible point in a 700 MHz licensee’s wireless network.

In late July 2007, Google urged the Commission to adopt rules for the auction that ensure that, regardless of who wins the spectrum at auction, consumers’ interests are served. Google also announced its intent to commit a minimum of $4.6 billion to bidding in the upcoming auction.

**A Partial Victory for Google and Third Party Apps**

Under the rules released by the FCC in early August 2007, Google had two important victories. Consumers will be able to use any software application or device on networks using a portion of the spectrum to be auctioned (points #1 and #2). However, the FCC did not approve the provision that would have required the winner of the auction to sell access to its network on a wholesale basis to other companies (point #3). This means that Google, or another company or coalition dedicated to open access, must win the auction in order to fully achieve open access. In the instance that a big telco wins the auction, they still have to leave it open to devices it does not approve or control. However, telcos are free to charge consumers or other third parties for access to its network (point #4).

While other telcos bid against Google, Google is partnering with T-Mobile and HTC on an upcoming G1 phone. To lure users away from ATT’s expensive iPhone data plan, T-Mobile is offering free G-mail to the first purchasers of the G1.

If all goes well for Google, it just may have an open (application/device) wireless network, a set of web-enabled software applications that run on any device, as well as a slick Smartphone of their own. It appears that Google has started the race to commoditize and abstract devices, operating systems and telco networks, which should have Microsoft and AT&T worried.

With a growing set of tools, such as Google Apps, Google has already started to abstract the operating system, which aims at the heart of Microsoft’s business model. For AT&T, their core business is the network. Controlling the network has given AT&T almost unrivaled power to dictate who has access to what wireless and broadband services. If Google ends up owning the 700 MHZ band, it will be position to abstract the telco networks as well.

Can Google do it? Can Apple? Will AT&T, Microsoft or IBM or someone else step in? At the moment, Google seems to be the best positioned given its current business model. Of course, it is not the only company that can embark on this initiative. The obvious other players in this high-stakes game would be the telco giants and software giants. It would be a bold move for Google and, arguably, the correct move in the long-term if trends toward open access continue to gain consumer favor.
The Perfect Storm: The FCC, Google, and Intelligent Agent Platforms

In light of the FCC auction and the anticipation of open access, a perfect storm is brewing that will enable the next-generation of location-based 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.

While revolutionary, such technology is just the tip of the iceberg when it comes to the type of intelligent content delivery that can occur on this type of platform. Think about personalized advertising pushed to individuals as they walk by stores, restaurants, coffee shops, concession stands, and souvenir shops in a mall, amusement park, resort, large stadium, or even a flea market. This would be entirely opt-in and configurable by the consumer, as they would be able to set preferences that filter out all extraneous information. The consumer can expedite their shopping process, and find the best deals on an item while stores can promote sales and inform consumers in real-time. Malls, parks, resorts or stadiums can differentiate themselves from competitors, as can the telco carriers, handset manufactures or any other company who dares to embrace this vision.

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.
Intelligent Mobile Platform
Requirements: A Top 10 List

While today’s location-based apps 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.

### Voyager™ ONE Software Abstraction Layer

**Existing Capabilities & Roadmap**

<table>
<thead>
<tr>
<th>JMS</th>
<th>MSMQ</th>
<th>DDS</th>
<th>JEE</th>
<th>NET</th>
<th>CORBA</th>
<th>SIP</th>
<th>UDDI</th>
<th>JTS</th>
<th>MTS</th>
<th>OTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messaging API</td>
<td>Security API</td>
<td>Discovery</td>
<td>Transaction API</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2P, P2-Group, P2-Enterprise, Enterprise-2P Messaging</td>
<td>P2P, P2-Group Messaging</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wired</td>
<td>WiFi</td>
<td>WiMax</td>
<td>UWB</td>
<td>GSM</td>
<td>CDMA</td>
<td>Bluetooth</td>
<td>Zigbee</td>
<td>NFC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Messaging API</td>
<td>Security API</td>
<td>Discovery (White/Yellow pages)</td>
<td>Transaction API</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protocols — SOAP, JOP, XML-RPC, RMI, Binary XML/Objects, TCP/IP, UDP, OBEX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Container &amp; Voyager Intelligent Agents</td>
<td>Ontology API</td>
<td>Database API</td>
<td>Rules API</td>
<td>Location API</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ontology Engine</td>
<td>Database Engine</td>
<td>Rules Engine</td>
<td>Location Engine</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Universal Collections &amp; Algorithms (Java, .NET, C++)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Android**, **JavaFX**
- **NET**, **Compact Framework**
- **Java (JSE, JME)**, **C++**, **Objective C**
- **Windows**, **Windows Mobile**, **Linux**, **Blackberry**, **Embedded OSs**, **Symbian**, **BREW**, **Real-time OSs**, **iPhone**

- Indicates Existing Capabilities
- Indicates 2008 Roadmap
- Indicates 2009 Roadmap
**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.
Privacy, Security, and Spam in a Networked World

When discussing a ubiquitously networked world one of the first concerns consumers raise is the protection of privacy. An extension of this issue is the risk of communications technology making consumers vulnerable to receiving a barrage of unwanted messages and advertising. There are several methods already in place to prevent this, including legislation and filtering controls, but the simplest way is to ensure messaging is an optional opt-in service for the consumer.

The Can Spam Act of 2005

When Congress passed the Can Spam Act in 2005, it became illegal in the US to send any message to a user without the user opting-in to receive such messages. While this law has slowed the proliferation of email spam generated in the US, it remains a huge problem. Enforcement will have similar challenges in a mobile situation. This is one argument for a single company, like Google, that will open up access to legitimate apps, while acting as a policing agent for unwanted advertising. Another argument proposes to keep communications software from going open source to control access from those who may abuse the pipeline.

Opt-In Profile

In support of the Can Spam Act, user’s participation can be on a strictly opt-in basis. A user would complete a detailed profile where they could specify what type of services and people they wish to contact them. Should a person’s preferences change his or her profile can easily be changed in real-time.

Location Filtering

Another way to combat random spam messaging is to confine messaging to the user’s immediate surroundings, therefore only messages from nearby people and businesses would be received by a user instead of a barrage of requests from non-relevant locations. The user’s profile could be set to receive only the most pertinent information using a predetermined radius that could be expanded if the user desires.

Security

In addition to standard network-level and payload level encryption, authentication or authorization, security software agents on the user’s device can watch for requests or data from nodes that are trusted or un-trusted. Agents also would enforce multiple levels of security that were sensitive to the type of information (e.g. payload) being passed (financial, business, social, recreational etc.), from and to whom, and even when and where. Again, users would be able to easily add and remove approved entities from their white list, just as they would any email subscriber, as well as add or remove locations and times they do not want to receive approved content.

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 Location-Based Apps

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);
   mobility.moveTo("/tokyo:9000");

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. turn it into an agent
   4. make it autonomous
   5. move it to another node

   // 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 to that community
   3. create another agent space on a remote node
   4. create another remote component and add it to 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/Subspace2");

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

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

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

   // enable web services support
   com.recursionsw.ve.web.services.WebServices.enableWebServices();

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

   // Create a client proxy
   XFireProxyFactory proxyFactory = new XFireProxyFactory();
   String hostName = InetAddress.getLocalHost().getHostName();
   IStockmarket market = (IStockmarket)proxyFactory.create(serviceModel, "http://dallas.recursionsw.com:8000/services/Stockmarket");

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

   // calls "news" method on every Consumer object in the community
   Multicast.invoke( subspace1, "news", new Object[]{ "newsflash 2!" }, "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" );
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 ONE 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

© 2007 Recursion Software, Inc.
2591 North Dallas Parkway
Suite 200
Frisco, TX 75034

972.731.8800
800.727.8674
recursionsw.com