www.psinaptic.com

PsiNaptic Inc.
1410, 10655 Southport RD SW
Calgary, Alberta
Canada T2W 4Y1
T: 403 720-2531
F: 403 720-2537

Java/Jini Solutions to USB and Bluetooth Profile Limitations

1. **With Java/Jini on the vehicle platform, ALL functionality of the portable device can be controlled and accessed from the vehicle control panel, irrespective of the underlying protocol.**

   Wired or wireless connections such as USB, Bluetooth, WiFi allows devices to connect and interact using protocols that specify functionality at the lowest common denominator. Bluetooth AVRCP for example allows for only basic control functions i.e. "volume control, start, stop, play, fast forward, rewind… ".

   Question is: ***will protocols and profiles be able to keep up with the plethora of new devices and services?***

   Portable devices streaming audio and video are already feature-rich, allowing the user to select artists, albums, favourites… etc., functionality not supported by the profiles and therefore not accessible by the vehicle.

   Managing, maintaining and supporting Bluetooth profiles, to keep up with all the new functions and features of mobile devices, will be very challenging and costly. OEM's can mitigate the development costs and time associated with creating, maintaining and validating/system testing multiple devices and their respective Bluetooth profiles, especially as the device manufacturers integrate new function/features. Because JMatos/CMatos utilize the standard and ubiquitous TCP/IP network stack for communication, application development for Bluetooth devices is significantly simplified. In addition, because JMatos/CMatos can offer an API implementation as Java byte code, the implementation can be downloaded and executed by a Jini client when needed. This simplifies application development for Bluetooth devices and future proofs applications that are no-longer dependent on profile specifications. With JMatos or CMatos and a PAN profile, you might not need any profiles! (More on this below)

2. **With Java/Jini the attributes and behaviour of the device can be downloaded as Java byte code and instantiated on the vehicle. The vehicle now knows everything it needs to know about the device and can control the device from the vehicle control panel.**

   Here's briefly how it works:

   - When a device using Jini enters a network and has services to offer, it will advertise these services through its Look Up Service (LUS). It will use a multicast annoucement to advertise its presence. Any device in a network that is looking for services will locate these LUSs and be able to search for the services it is looking for. When it finds the ones it wants, it will acquire the service (which can be a driver, GUI or application) from the LUS and run it locally on its JVM. The applications will then perform the intended functions.

- Whenever a service or device leaves the network or is no longer available, the service object and drivers are automatically removed from the memory of the device that was using the service, leaving the memory available for other functions. A leasing mechanism is used to accomplish this. Jini also provides event notification for added flexibility in communications between applications.

3. **Alternatively, a CMatos solution in the vehicle does not require a JVM.**

   One alternative to a full Java/Jini implementation on the vehicle platform is to have a CMatos solution. CMatos is an implementation of the Jini LUS and related protocols that enable very small-embedded non-Java processors to offer Java based services to any computing device that acts as a Jini Client. What this means is that the vehicle can offer its services (audio, LCD, handsfree, diagnostic etc.) to any device (PDA, MP3, medical, navigation, POS) that acts as a Jini client. The device is going to look for, discover, and download the services that it needs and then utilize the vehicle services. For example, if a vehicle has a large LCD, it can offer its display as a service to a PDA running a navigational application. The driver will view navigational information on the LCD screen, instead of on the PDA.

4. **Java/Jini defines device functionality at an API level and not the protocol level, significantly simplifying specifications for discovery, connectivity and interaction.**

   Ease of specification is one of the greatest advantages of Jini. A Java interface is much simpler (and quicker) to specify than a Bluetooth profile. The behavior of the Jini service (driver, GUI or application) on the client does not need to be specified as it can be supplied by downloadable Java byte code from the device when needed. **Jini + PAN profile will significantly reduce the need for Bluetooth profiles.** This will greatly reduce time-to-market for new applications such as games, medical and Point of Sale machines for example.

5. **Java/Jini enables PAN, WAN, LAN Network visibility.**

   Jini services are visible and accessible to the larger TCP/IP network. The TCP/IP network may be much larger than a Bluetooth PAN. A profiles-based service will not be visible beyond the Bluetooth piconet. Java/Jini enables applications on a device to interact with other applications and devices across PAN, WAN, LAN networks and can therefore:

   a. Enable the seamless discovery and integration of mobile devices into the vehicle (mobile phone, MP3 Player, medical device, game player or PDA). Controlling mobile devices coming into the vehicle and potentially linking them to back-end e-business infrastructure - allowing anytime, anyplace access to information/services.
   b. Enable vehicle centric information (i.e. diagnostic) to be accessed/shared by the user or service station
   c. Enable the vehicle to e-transact etc... From back-office to the automobile platform to enable the installing or un-installing bundles

6. **Java/Jini code mobility future proof's devices and applications.**

Currently, in order for there to be any kind of interaction such as:

a. device and vehicle (for vehicle diagnostic, data, voice, music, video, or navigation, tracking) or
b. vehicle and off-board or distributed applications (service station, navigational etc)

make, model, revision level specific software needs to be pre-installed on both the device and the vehicle. With a plethora of new consumer electronic devices coming onto the market every 9 to 18 months, its impossible for OEM's vehicle platforms to have make, model and revision level software pre-installed in the vehicle, or conversely for users to be techno-aware and pre-install OEM's software onto their devices. This problem is further aggravated with Bluetooth profiles that are still mired with interoperability issues and generally lags the consumer electronic industry.

Jini is a comprehensive distributed computing framework using the full capabilities of Java providing a large suite of options for a system designer and for the applications. It will operate well on very small networks and excel in very large networks. It is very robust and is ideal for ad-hoc IP based networking for mobile devices. Probably the most important component to meet the goal of robust distributed computing on a network is the concept of providing code mobility between devices and application spaces. **One of the key benefits of this code mobility is future proofing applications and devices.**

JMatos/CMatos ensures that the device gets the right software only when it needs it. This allows a user to consume a service in an ad hoc manner without having to install and maintain specific software on their device. Currently the client software must be specifically written and installed for each type/kind/model of device, limiting its capability.

JMatos/CMatos leverages the ability of Java and the JVM to 'write once run anywhere', by providing the actual byte-code required by the client. This byte-code (or Object) is loaded and instantiated on the client and run locally.

7. **Open the door to third party software applications.**

It's still easy to think about integrating an iPod or Hands Free Module in the traditional sense - using proprietary software, cradles and connectors. But long term, proprietary connectivity software limits applications.

On the other hand, it's difficult to imagine an environment in which services are consumed or offered in a disbursed, ad-hoc manner. But if you think about it, Java is taking off like a horse on fire and "leased" application can be downloaded. Off-board or distributed navigation application can be downloaded and used temporarily such as when the user goes on a trip to an unfamiliar location. Other applications might be concierge services or points-of-interest. JMatos on the OEM's telematics platform can enable this capability.

There are already 2.5 Billion Java enabled devices (700 million phones!). In 2-4 years from now, there will be many tens of billions wireless, Java/Jini enabled devices in response to consumers desire for untethered lifestyles, with gadgets blurring lines between work and play, connecting seamlessly to PAN's, LAN's and WAN's accessing everything digital - voice, data, video.

Question is: will OEM's be actively pursuing platform independence, scalability, maintenance, smart downloads and significant reduction in software development?

8. Cost savings

Many OEM's are looking to change their in-vehicle platform from static, single purpose to dynamic, multi-tenant environment, in a controlled and safe manner. With Java/Jini 30-50% productivity gains can be expected on the development platform due to modularization, scalability, priority execution and life cycle management, as well as amortizing the code over the same processor.

OEM's could be first to establish a controlled access proxy to the developer community and monetize the platform; building an ecosystem for the Java community to build applications; having a more capable consumer platform by being an external interface provider exposing the vehicle interface to mobile devices; using Java/Jini to future proof the platform and enable secure over the air provisioning...

Multi-media becomes an affordable option in entry level vehicles and late integration of devices and technologies into product development and deployment cycle becomes possible.

Less memory required by devices (thanks to auto cleanup).

Much less (if any) user input (thanks to auto load and run).

Many more services available as a result of services discovered and used dynamically beyond what has been loaded and is resident in the device.

The ability to differentiate manufacturers whose user interfaces can be downloaded dynamically.

Automatic revisions and upgrades (because the mobile device will deliver the latest version of the software that it knows about and/or the vehicle will be dynamically connecting to a LAN/WAN for updates).

Reduce development costs and effort (thanks to ability to leverage proven Java building blocks and Jini technology).

Quicker prototype development cycles. Faster integration of new services and devices into the vehicle development cycle at minimal cost.

Reduced deployment and maintenance costs - Unlimited applications.

Improved revenue opportunities for vehicle manufacturers using the architecture

9. User Experience

Impact to the user experience: JMatos/CMatos can enhance the consumer experience while still meeting the changing needs of mobile user because it enables seamless and dynamic interaction between devices and the vehicle, without human intervention or pre-installed software. A device, user or vehicle should be able to move from environment to environment (i.e. home, office, vehicle) and either offer or consume a service, on an ad hoc basis.

There are many challenges to this vision. Having pre-installed software for each service, each device, each model is simply not feasible.

10. Standards

JMatos/CMatos software technology is based on standard Java, widely adopted across all functional areas: mobile, vehicle, medical, home etc.

Ubiquitous approach diametrically opposite to typical solutions which tend to be proprietary, or make/model product specific, or still involve cradles, cables and connectors.

ADDENUM

1. **Review Jini Networking technology**
2. **A case for Java/Jini**
3. **JMatos/CMatos and Bluetooth Profiles**

1. Review Jini Networking technology

OVERVIEW

Jini network technology provides a simple infrastructure for delivering services in a network and for creating spontaneous interaction between programs that use these services regardless of their hardware/software implementation.

Any kind of network made up of services (applications, databases, servers, devices, information systems, mobile appliances, storage, printers, etc.) and clients (requesters of services) of those services can be easily assembled, disassembled, and maintained on the network using Jini Technology. Services can be added or removed from the network, and new clients can find existing services - all without administration.

TECHNOLOGY

- **Technology Advantage**
  Jini technology is truly open to the marketplace, allowing more widespread innovation and providing an open standards based environment for developers, service providers and content creators alike.
- **Operational Advantage**
  Jini technology enables open end-to-end solutions for creating dynamically networked products, services, and applications that scale from device to the enterprise.
- **Competitive Advantage**
  Jini technology solutions enable developers, service providers and content creators to

gain a competitive business advantage and capitalize on new revenue streams in the service-driven network, by rapidly and cost effectively developing and deploying compelling new applications and services for their customers worldwide.

Network Evolution

Over the last quarter century, network technology has evolved rapidly and in some unexpected ways. Client/server and multi-tier models operating within a single business enterprise have given way to an Internet/Web environment where services are provided by nodes scattered over a far-flung network.

Today, the next generation of network interaction is emerging that will place unprecedented demands upon existing network technologies and architectures. For example, participants in one network will need to directly access and use the services provided by participants in another network. It is in this network environment - one of mind-numbing complexity driven by geometric increases in scale, rate of change, and multiplicity of participant interactions - that the simplicity of the Jini architecture clearly wins.

How Jini Technology Works

By using objects that move around the network, the Jini architecture makes each service, as well as the entire network of services, adaptable to changes in the network. The Jini architecture specifies a way for clients and services to find each other on the network and to work together to get a task accomplished. Service providers supply clients with portable Java technology-based objects that give the client access to the service. This network interaction can use any type of networking technology such as RMI, CORBA, or SOAP, because the client only sees the Java technology-based object provided by the service and, subsequently, all network communication is confined to that Java object and the service from whence it came.

When a service joins a network of Jini technology-enabled services and/or devices, it advertises itself by publishing a Java technology-based object that implements the service API. This object's implementation can work in any way the service chooses. The client finds services by looking for an object that supports the API. When it gets the service's published object, it will download any code it needs in order to talk to the service, thereby learning how to talk to the particular service implementation via the API. The programmer who implements the service chooses how to translate an API request into bits on the wire using RMI, CORBA, XML, or a private protocol.

IN SUMMARY

Jini is a comprehensive networking infrastructure using the full capabilities of Java providing a large suite of options for a system designer and for the applications. It will operate well on very small networks and excel in very large networks. It is very robust and is ideal for mobile networks. Probably the most important component to meet the goal of robust distributed computing on a network is the concept of providing code mobility between devices and application spaces. **One of the key benefits of this code mobility is future proofing applications and devices.**

2. A case for Java/Jini

Why the Java Platform?

(note: all increase are since June 04)

2.5 Billion devices (total) - 42% increase

1B Java Cards! - 67% increase

708M Java Powered phones (Source: Ovum) - 23% increase

700M PCs with Java - 8% increase

140+ carriers - 50% increase

912 JCP Members - 14% increase

4.5M developers - 12% increase

Secure	
Portable	
Network enabled	Dynamic software upgrades Able to execute secure 3rd party code - expose specific elements to mobile devices (API predefined interface)
Pervasive	Open standard Being the first to establish a controlled access proxy to the developer community and monetize the platform; building an ecosystem for the Java community to build applications; having a more capable consumer platform by being an external interface provider exposing the vehicle interface to mobile devices; using Java/Jini to future proof the platform and enable secure over the air provisioning...

Cost effective	In a multi-tenant environment, 30-50% productivity gains on the development platform due to modularization, scalability, priority execution and life cycle management, as well as amortizing the code over the same processor. Multi-media becomes an affordable option in entry level vehicles and late integration of devices and technologies into product development and deployment cycle becomes possible.

Why JMatos/CMatos Jini Network Technology?

Why use JMatos/CMatos Jini Technology?	Provides an environment for creating dynamically networked components, applications, and services that scale from the device to the enterprise. Offers an open development environment for creative collaboration through the Jini Community.
Unique Qualities	Code Mobility: Extends the Java programming model to the network; i.e., moves data and executables via a Java object over a network Protocol agnostic: Provides the ultimate in design flexibility Leasing: Enables network self-healing and self-configuration; i.e. improving fault tolerance
Unique Benefits	Resiliency - Networks readily adapt to changes in the computing environment Integration - Allows fast, easy incorporation of legacy, current, and future network components
The use of JMatos or CMatos should result in:	- Less memory required by a device (thanks to auto cleanup)

	- Much less (if any) user input (thanks to auto load and run) - Many more services available as a result of services discovered and used dynamically beyond what has been loaded and is resident in the device - The ability to differentiate manufacturers whose user interfaces can be downloaded dynamically - Automatic revisions and upgrades - Reduced development costs and effort - Reduced deployment and maintenance costs - Unlimited applications

3. JMatos/CMatos and Bluetooth

Jini is a very attractive networking technology to replace the plethora of Bluetooth profiles and attendant inter-operability issues. Bluetooth phones today support the standard profiles (HSP, HFP, DUN server, OPP, and maybe Sync) and will support the emerging profiles (PBAP, BMP, and PAN). Smartphones, PDA's and stand-alone digital audio players (DAPs) will support the streaming audio profiles (A2DP and AVRCP). This is what we know today. Is there going to be profiles for every Point of Sale machine or medical device? Managing, maintaining and supporting Bluetooth profiles will be very challenging

At the application level, JMatos/CMatos and the PAN profile provides the ability for devices to discover each other and/or offer services over it's TCP/IP network. Devices offer services to the network over the PAN profile. Other devices can locate these services and use them. This can be done without the need to define and agree on profiles for each application. An API is all that is required as a definition for each application. The discovery, offering and using of services is network independent and is not constrained to Bluetooth or the piconet, but will operate smoothly across all TCP/IP networks.

Functional requirements are:

- Every Bluetooth device that wishes to benefit from and participate in a Jini federation will need PAN profile with multicast.
- Every device that wishes to offer services and does not have a Java Virtual Machine (JVM) will have to carry a Look-Up Service (LUS), around 40 kb. The LUS is the critical

part of Jini where a service is registered to be available. Jini encourages multiple LUSs, and so each device could carry its own LUS with its own intrinsic services. These services will use Java byte code as the language and method to exchange drivers and data and communications/protocols between applications.

- Any Bluetooth device wanting to use a service will require PAN profile with multicast and a JVM. These are not required to have a LUS.

Behaviour of Application and Devices:

- When a device using Jini enters a network and has services to offer, it will advertise these services through its LUS. It will use a multicast annoucement to advertise its presence. Any device in a network that is looking for services will locate these LUSs and be able to search for the services it is looking for. When it finds the ones it wants, it will acquire the driver from the LUS and run it locally on its JVM. The applications will then perform the intended functions.

- Whenever a service or device leaves the network or is no longer available, the service object and drivers are automatically removed from the memory of the device that was using the service, leaving the memory available for other functions. A leasing mechanism is used to accomplish this. Jini also provides event notification for added flexibility in communications between applications.