An Oracle White Paper
March 2011

Making the Smart Grid Smarter
with Embedded Java
Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
# Table of Contents

Executive Overview...........................................................................................................4  
Who Should Read This White Paper .................................................................4  
Introduction ..........................................................................................................................5  
Situation Analysis .............................................................................................................5  
Crisis and Opportunity .....................................................................................................5  
Oracle and the Smart Grid ...............................................................................................6  
Vision – The Emerging Smart Grid and the Role of Java ..............................................6  
Smart Grid Architecture – End-to-End ...........................................................................6  
  The Last Mile – Delivery and Consumption .................................................................6  
  Generation, Grid Management and Transmission .........................................................7  
  The Energy “Back Office” ...............................................................................................7  
  Premises Devices ............................................................................................................8  
The Smart Grid Ecosystem ...............................................................................................8  
Technology Enablers ........................................................................................................9  
Challenges to Smart Grid Implementation ......................................................................9  
Java – A Platform for Energy Intelligence ......................................................................10  
Benefits of a Java-based Smart Grid .............................................................................10  
Java Technology in Focus ...............................................................................................11  
  Java Profiles for Smart Energy ......................................................................................11  
  A Java-based Smart Grid S/W Architecture .................................................................11  
  User Interface ................................................................................................................14  
  Hardware Platform Support ..........................................................................................14  
  Java vs. Android for Smart Grid Applications ..............................................................15  
Remote Administration and Device Management ..........................................................16  
  OSGi .................................................................................................................................16  
Conclusion ........................................................................................................................17  
Smart Grid Resources for Java .......................................................................................17  
  Java Platform Resources ...............................................................................................17  
  Smart Grid Resources ....................................................................................................17
Executive Overview

This white paper examines the challenges of building of a truly “smart” smart grid and the benefits of using Oracle Embedded Java platforms in that build-out. In the context of government and business responses to energy supply shortfalls, it highlights the ecosystem that is evolving to bridge energy generation, delivery and conservation gaps. In particular, it focuses on role of Java in the “last mile” of electricity delivery, including smart meters and smart-meter infrastructure outside the premises, and energy gateways, displays and smart appliances on the inside.

Who Should Read This White Paper

This white paper should be of interest to anyone interested in the emerging Smart Grid, but is directed in particular to technical resources and business people responsible for Smart Grid rollout and for creating energy distribution and management equipment:

- Solutions Architects, IT Directors and Chief Architects
- Integrators and OEMs
- IT staff and physical plant managers in enterprise and government
Introduction

Situation Analysis

After decades of “all you can eat” electrical energy consumption, governments and power companies find themselves facing a set of intertwined challenges:

- Increased pressure on global fossil fuel supplies and pollution and global warming impact from using still plentiful fuels like coal
- Regulatory restrictions, high costs and NIMBY (Not In My Back Yard) constraints and increasing generation capacity, especially for nuclear power
- Aging distribution infrastructure (the legacy grid)
- Increasing energy demand, from both developed and developing countries, from economic growth and from anticipated demand from charging electric vehicles (EVs)
- Limited ability to monitor and respond to changing energy demand in real-time
- Security threats to the distribution grid and to consumer privacy

Over the last decade, the confluence of these factors has lead to climbing energy costs, power quality degradation, brownouts and blackouts. The unpredictable nature of energy supply and demand, beyond its palpable impact on the lives of consumers, creates untenable complications for all types of business, from manufacturing to services industries.

Crisis and Opportunity

This burgeoning crisis has different responses from government and industry. Governments around the globe, from the US to Germany to Brazil, are investing at many levels, starting with programs focused on improving infrastructure, and also providing indirect (tax-based) and direct incentives for innovation (grants, etc.).

Industries beyond “big energy” are responding as well, from IT to consumer electronics to manufacturing and industrial control. Impetus comes both from government stimulus and from market opportunities stemming from the crisis itself.

Entrepreneurial opportunities take two primary forms:

- Technology to monitor and streamline generation, distribution, delivery and consumption
- New businesses and business models arising from those technologies

This white paper focuses on the technical and business options, and rationale, for embedding intelligence into the grid for delivery and for attenuating consumption. In particular, it highlights using Java to make legacy and emerging electrical distribution smarter and to imbue premises devices that consume energy delivered to them.
Oracle and the Smart Grid

Vision – The Emerging Smart Grid and the Role of Java

Oracle believes that the key to making the Smart Grid smarter lies in two development trends:

- Adding energy awareness to all types of intelligent devices
- Making intelligent energy monitoring and control part a de facto part of homes, offices and other places of businesses

*Energy awareness* means that embedded designers integrate energy efficiency and management into the hardware and software of intelligent devices – home appliances, entertainment systems, office equipment, environmental controls – essentially any programmable system.

Intelligent energy monitoring/control is embodied in systems and software that measure actual energy usage by communicating with the universe of devices in the home and workplace.

To enable efficient operation, communication, monitoring and control, developers need to rethink embedded systems as more than single-purpose devices and must instead design and build applications platforms, systems whose software can respond to changing local conditions, and through software upgrades and additions, respond to changing market economics, especially for energy.

With increasing deployment of 16 and 32-bit embedded processors and microcontrollers, an increasingly large swath of intelligent devices can today run applications, distinct from the software platform itself. Embedded OSEs – Linux, Android, embedded Windows and advanced RTOSes (real-time operating systems) – support deployment of recognizable shrink-wrap-type applications as on the desktop and in the data center. Device manufacturers (OEMs) typically install this software at the factory (Pre-Load), but following the model of mobile apps stores, devices of all kinds are beginning to support after-market (Post-Load) application deployment as well.

Smart Grid Architecture – End-to-End

In theory, the Smart Grid encompasses every aspect of energy generation, delivery and consumption. However, upstream investments in generation and transmission will yield limited results without changes consumption patterns on business and consumers premises.

The Last Mile – Delivery and Consumption

This Last Mile (or last ten yards) is the focus of this white paper. This includes nodes that use electricity directly, monitor overall consumption, and reside in or near homes, offices and other premises.

- Data aggregation devices – smart meter readers/concentrators/routers and microgrid controllers, typically located on power poles
- Smart Meters
- Energy Gateways (HEG) and Premises Displays (IHD)
- Smart appliances and other premises equipment
Generation, Grid Management and Transmission

The larger “macro” grid that entails energy generation and transmission, and the management of that infrastructure also benefit from embedded intelligence supplied by Java. The myriad control and monitoring systems in modern power plants, high-tension distribution, power substations and other heavy infrastructure today reflect multiple generations of uncoordinated, semi-standardized electrical, mechanical and software engineering efforts.

Standardizing on embedded Java as the platform for these nodes offers developers and utilities

- Interoperability and code reuse across grid infrastructure
- Multiple options for machine-to-machine (M2) interaction across the grid, end-to-end, facilitating load distribution, monitoring and switching
- Remote monitoring/reporting on (headless) node status using secure APIs and Java-based web application components
- Real-time performance for low-latency monitoring of electrical behavior
- Richer, standards-based data streams to facilitate node availability and graceful bring-up and shut-down of grid nodes and elements
- Intelligence to manage the legacy grid better and move incrementally to lowering the carbon footprint of industrial scale power generation and delivery
- A platform for ongoing incremental improvement by managing and updating deployed nodes to meet the needs of an increasing smart grid

The Energy “Back Office”

Oracle also offers Java-based technology and other solutions for other Smart Grid nodes, especially for utility customer relationship management and business intelligence. Learn more at http://www.oracle.com/us/industries/utilities/

Figure 1. – The Smart Grid, End-to-End
Premises Devices

Terms like HEG (Home Energy Gateway), IHD (In-Home Display) and HAN (Home Area Networks) speak to the home/consumer marketplace. However, the same nodes, technologies and challenges apply equally to small and medium business premises (SMB), to enterprise branch offices, and also to multi-dwelling premises (condominiums, apartment houses, etc.).

The HAN is actually the sum of several networks, encompassing existing LANs (Ethernet and WiFi) and networks specifically designed for premises automation and monitoring (Zigbee, Z-Wave, X10, power-line communications, etc.).

- **WiFi** – standard 802.11b/g/n wireless networking, connecting the HEG with local computers and tablets, and Internet backhaul
- **Ethernet** – standard wire line connection to LAN and Internet backhaul
- **Bluetooth** – peripherals and mobile devices, both as interfaces to HEG and for energy management and control
- **Zigbee** – a standards-based (IEEE 802.15.4-2003) wireless protocol for industrial and home networking with profiles for energy monitoring and control
- **Z-Wave** – a proprietary wireless networking protocol for home automation
- **HomePlug** – a power-line communications (PLC) protocol for home/SOHO
- **X10** – legacy power line and wireless home automation network

Figure 2. – The multiprotocol HAN for Communications, Energy Monitoring and Control

The Smart Grid Ecosystem

Figure 1. is populated with and serviced by a rich ecosystem of technology, product and services suppliers. These participants include

- Device manufacturers (OEMs), from Consumer Electronics companies to networking equipment providers (NEPs) to home automation, building managed appliances (washers/dryers, refrigerators, HVAC, electro-domestics, lighting, etc.) and devices to manage them (IHD/HEG)
- Energy industry OEMs manufacturing smart meters, pole-top systems, solar panels and other vertically targeted equipment
- Power generation utilities
- Software and services companies serving the above
Technology Enablers

The long-term evolution of energy distribution follows a path from electro-mechanical to linear/analog to digital to embedded intelligence. That is, from systems focused solely on moving electrons to ones that participate actively in energy monitoring and consumption control.

Such intelligent devices depend upon a handful of technical enablers that include

- 16 and 32 bit microprocessors and SoCs (systems-on-a-chip) based ARM, Atom, MIPS and other architectures
- Embedded storage technology – HDD and flash memory
- TCP/IP networking
- HAN protocols – Zigbee, Z-Wave, PLC, etc. – and devices enabled for them
- Programmable and intelligent applications platforms, Java and Linux in particular

Challenges to Smart Grid Implementation

Bringing national and international energy grids – often 100 year-old infrastructure – into the Twenty-first Century presents challenges that span generation to transmission to distribution to consumption. These challenges include

- **Legacy systems and mindset** – large-scale energy distribution began in the late 19th Century and a large part of the energy business and infrastructure remains little changed since then. A truly smart Smart Grid requires rethinking how energy is supplied and consumed, from a static “electro-mechanical” view to one built of smart application-centric embedded computing
- **Regulatory regimes and energy distribution culture** – the particulars of the relationships among government and generators greatly impact how energy is delivered today and limit the latitude and reach of smart energy
- **Equipment life-cycles** – key premises equipment and distribution infrastructure have long product life-times and acquisition cycles mismatched with short-term needs to conserve
- **Premises and poll-top provisioning** – regulatory strictures, right-of-way and logistics can present formidable barriers, sometimes defying common sense and business sensibilities of new entrants in the energy businesses
- **Reliability, service and support** – monitoring energy use and controlling final distribution and use constitute not just a business-critical activity, but a life-critical one as well: faulty and difficult-to-use equipment and unforeseen outages of electricity, heating and air conditioning can threaten not just people’s livelihoods, but their lives as well
- **Information backhaul** – the flow of energy and information across the legacy grid is narrow and asymmetrical: power flows outward and usage information inward at human pace from meter
readers and handwritten reports. A modern smart grid needs multiple, interactive communications channels for usage, billing, service and demand information

- **Consumer buy-in** – while utilities have long-standing and nearly exclusive relationships with energy consumers, energy suppliers need to improve communication with their customers and educate them on the benefits of energy efficiency, tiered billing rates and demand response.

- **Security and privacy** – a connected and intelligent energy grid, abuzz with usage statistics, demand messages and billing information, provides an attractive target for crackers, phishers, identity thieves and even terrorists.

---

**Java – A Platform for Energy Intelligence**

In this complex and challenging environment, Java presents developers and integrators with a familiar and flexible tool to meet multiple, diverse energy ecosystem requirements.

**Benefits of a Java-based Smart Grid**

- **Security** – Java has an excellent track record in providing a secure applications platform, for embedding in Smart Grid devices and also for desktop and data center deployment. Java profiles (e.g., Java SE Embedded) offer developers rich APIs (java.security, etc.) for encryption, authentication, key management, etc.

- **Performance** – Java byte-code based execution often leads to question about throughput and responsiveness of Java-based applications. In the fifteen-year history of Java, Oracle and the Java developer community have invested hundreds of man-years in optimizing Java virtual machine and JIT (Just-In-Time compiler) performance for different use cases. The result is that today, Java performance exists on par with traditional compiled C and C++, with lightning fast throughput and low-latency response for real-time systems, supporting Smart Grid applications ranging from monitoring/controlling mesh-networked appliances to high-speed data logging to end-to-end demand response.

- **Reliability** – the Java platform is employed in all types of information technology, from enterprise to desktop to embedded systems. Its broad use and worldwide development community have yielded an extremely reliable platform, and the syntax and semantics of the Java language help developers build and deliver inherently more reliable applications. In studies of Java application code and of underlying JVM code and class libraries, Java reliability stands out over comparable C/C++ implementations and run-times.

- **Interoperability** – the long-time rallying cry of Java is “Write Once, Run Anywhere” (WORA). The core Java architecture and profiles (see below) and the Java Community Process are key to ensuring maximum interoperability among Java implementations. Moreover, Java applications can also interoperate with software and protocols outside the current Java corpus, through the Java Native Interface (JNI) and through available bindings to libraries and software stacks written in C, C++, assembler and other run-time environments, especially for Smart Grid applications.
**Code Re-use** – Java offers two levels of code re-use: source and byte-code. Java source code is eminently portable and the language and profile definitions highly stable. Byte-code – Java compiled to an abstract machine instruction set – is by definition transparently portable across JVMs and underlying microprocessor instruction sets. The combination of portability and interoperability ensures one of the highest indices of code re-use of any embedded development tool.

**End-to-End** – Java code and expertise can be applied all the way across the emerging Smart Grid, from generation to delivery and from consumption to conservation. While the obvious locales for Java deployment are in user-facing devices with graphical UIs, Java is also an excellent choice for headless control and server-type devices across the Smart Grid, with fast throughput and strong real-time responsiveness.

**Longevity** – the evolution of Java rests in the hands of a worldwide developer/user community, Oracle Corporation, a leading Fortune 50 company, the Oracle partner and customer ecosystem, and hundreds of other substantial enterprises. The long-term commitment of both community and corporate interests in Java have sustained the platform for almost two decades and will continue to do so well into the future.

### Java Technology in Focus

#### Java Profiles for Smart Energy

In Java’s 15-year deployment history, market and application requirements led to development of multiple Java profiles – collections of APIs (classes) optimized for different use cases. These profiles include Java Card, Java ME, Java SE, and Java EE. These profiles map to Smart Grid applications as follows:

- **Java ME** – for intelligent appliances, communications, aggregation, control and monitoring systems
- **Java SE** – for high-performance IHD and HEG devices and data concentrators requiring higher throughput, integrated storage and display capabilities. Oracle also offers Java SE Embedded, optimized for resource-constrained applications.
- **Java Card** – for resource-limited applications, including many Smart Meters, remote controls and low-end premises equipment – simple appliances, electrical equipment, etc.

#### A Java-based Smart Grid S/W Architecture

This white paper focuses on four Smart Grid node types: data concentrators, smart meters, IHD/HEGs and smart appliances / premises devices. Figure 3. outlines key software components in a base stack to address these four device categories:
Let’s briefly examine these divergent device types, their requirements, and how they leverage Java capabilities for Smart Grid applications:

**Data Concentrator** – these headless (Pole-top Systems) systems primarily act as routers and access points that let Smart Meters “phone home” periodically with usage data, and support commands for provisioning and service disconnect. Embedded Java and storage/db capabilities gives these systems options for local caching and analysis of client smart meter data, especially for maintenance and trouble-shooting purposes. Java’s inherent scalability and performance also extends the lifetime of these systems, allowing them to provision and manage a large number of Smart Meter client devices that can grow over time. These systems need to communicate with Smart Meters over a number of channels, and due to their strategic role in provisioning and relaying customer energy usage data, also benefit from Java security.

Utilities and premises owners also deploy these types of systems as controllers\(^1\) for autonomous microgrids that generate local power and act as backups against grid-wide outages.

**Smart Meters** – these devices today represent a large and growing revenue opportunity for energy sector OEMs. Currently, Smart Meter rollout parameters constrain the functionality of these devices to monitoring and reporting energy usage, they present interesting opportunities for accelerating demand response and participating in HAN monitoring and control activities (across the premises demarcation). As such, they need protocols and intelligence for upstream and downstream communications – to poll-top systems, to HEG/IHDs, and to select high-load HAN nodes, especially HVAC, pool pumps and hybrid/electric car chargers. The evolving role of the Smart Meter also benefits from Java’s programmable platform functionality: relatively “dumb” Smart Meters at rollout can easily be re-provisioned for more comprehensive capabilities as legal and regulatory environments change.

\(^1\) E.g., the Echelon Edge Controller Node – [http://www.echelon.com/metering/ecn.htm](http://www.echelon.com/metering/ecn.htm)
**IHD/HEGs** – In-Home Displays and Home Energy Gateways hold the greatest opportunities for Java deployment in the Smart Grid. The IHD/HEG needs to manage and control a multi-faceted and always changing environment. Today, penetration of energy-aware appliance is quite low. In the next several years, mandates for energy-efficient white goods in the U.S. and abroad, will boost presence of smart appliances in the market and on premises. Over the next decade, homeowners will upgrade aging appliances and add new ones to their homes.

The broader vision of IHD and HEG-type devices places them at the center of the energy-efficient home. Java is the ideal platform to support this central, user-interactive role. It is also an ideal applications platform to accommodate appliance-specific applications and widgets and third-party apps that extend the pre-load functionality of IHD and HEG hardware with after-market post-load plug-ins.

As such, IHD/HEG systems will leverage the most functionality from the underlying Java-based stack, needing to support multiple communications and HAN protocols, offer users interfaces locally and via the Cloud, support downloaded and over-the-air software upgrades, and do so with top-notch data and operational security.

**Smart Appliances** – many appliances and devices in the prototypical HAN exhibit constrained functionality and BOM/market price points – light switches, power strips, electric heaters, smoke detectors and myriad other devices with simple operational states, limited or no user interface and liberal or non-existent latency/response requirements. These devices are probably best served by microcontrollers with no real s/w platforms or low-cost multi-functional units running Java Card. However, there are many categories of “white goods” and other household systems that would greatly benefit from inclusion of intelligence conferred by embedded Java.

Examples include

- Internet connected HVAC systems and other environmental controls
- Smart refrigerator/freezer units, with capabilities to track contents/expiration and deliver content through embedded displays (recipes, nutritional info, etc.)
- High-wattage internet-connected audio-visual equipment – flat screen TVs, DVRs, STBs, etc.

For these and comparable appliances, embedded Java enables richer base functionality without margin-sapping BoM impact. Embedded Java also enables these devices to participate in HAN-based energy management by point-casting more comprehensive energy profile information and allowing optimization of energy use harmonized with smart-meter multi-tier billing: for example, freezers running high-current defrost cycles when energy costs are low and saving power when the premises are empty and no one is likely to open the door. At a minimum, smarter appliances would need Java ME, HAN protocol stacks, and application-specific device drivers to run relays, sense temperature, etc.

Internet-enabled appliances would also require TCP/IP, and higher-end devices code to support a device user interface (see below).
User Interface

A Java-based system architecture gives developers and integrators maximum flexibility in specifying and building user interfaces for Smart Energy devices, with multiple roles for Java on HEG/IHD, appliances, smart meters, and data aggregation systems.

**Local Display** – an “In-Home Display” (IHD), by definition, includes a display, from small, simple numerical LCDs output to full-featured color LCD and touchscreen input/output. Java Swing and the Light Weight UI Toolkit (LWUIT) simplify prototyping and building user-friendly interfaces for Java applications.

**Browser UI** – to reduce BoM cost for HEG devices, and to provide UIs for other Smart Grid nodes (appliances, smart meters, pole-top systems, etc.), a headless Java-based Smart Energy device can easily present a web-based user interface for display on PCs, notebooks, tablets and other client devices over a network. Java provides a rich set of resources for building embedded web servers (e.g., com.sun.net.httpserver) and can also form the basis of the client-side UI with Java Applets.

**Mobile Apps** – increasingly ubiquitous smartphones and requirements to control Smart Grid devices remotely predicate building mobile applications for secondary (or primary) Smart Energy control and monitoring. A mobile app can just encapsulate a Browser UI in an application wrapper or can have a “life of its own”, with capabilities unique to mobile host platforms (input methods, location info, etc.). Over three billion phones already deploy Java, making Java an ideal tool for creating portable, multiplatform mobile apps. Note that unless an app is designed strictly for LAN-based monitoring and control (usually via WiFi), the mobile app architecture will have to include access to the Smart Energy device through some kind of gateway, usually also amenable to Java programming.

Hardware Platform Support

From its inception, Java design and implementation have emphasized platform independence. By hosting Java applications in virtual machine environments (JVMs), Oracle and the universe of Java licensees have enabled migration of Java across systems of all types, from enterprise servers and blades
to desktop computers to network infrastructure to myriad mobile and embedded devices. Today, Java programs can run on microcontrollers and mainframes and Java run-time environments exist for operating systems that include Microsoft Windows, MacOS, Linux and several dozen embedded and real-time OSes\(^2\).

Embedded systems designers enjoy nearly universal availability of Java across 16, 32 and 64-bit processors from all major architecture families – ARM, Intel Architecture (x86/IA), MIPS, Power Architecture, SuperH and application-specific silicon.

Java vs. Android for Smart Grid Applications

Many of the arguments in this white paper for Smart Grid build out using Java would appear to apply equally to Google’s Android platform. Both Java and Android present OEMs, developers, integrators and users with applications platforms. Both present developers with an object-oriented programming environment and integrated development tools. But the similarity ends there:

- Android Dalvik is not a certified Java platform, requiring recoding and rebuilding of ubiquitous enterprise, desktop and embedded Java code for use in Smart Energy applications. With embedded Java, legacy Java code just works
- Java platform performance and responsiveness outstrips and outshine that of Android – learn more at [Java SE Embedded Performance Versus Android](#)
- Java has been retargeted to every embedded platform of merit, from ARM to MIPS to Power Architecture and beyond, and across Intel and AMD desktop systems, blades and servers. Android supports select ARM-based SoCs, but does not officially or effectively support any other hardware platforms (reference ports to x86 are not deployment-ready)
- Java is architecture for maximum portability – WORA is more than just a slogan. While Android builds in a “Java-like” Dalvik platform, use of native code interfaces is so pervasive that a large proportion of Android apps won’t run (or won’t run well) across different versions and implementations of Android, and won’t run at all on emerging Android for MIPS, Power Architecture and other CPUs. Given that MIPS and Intel Architecture (Atom in particular) are the main contenders for many Smart Grid applications, Java offers OEMs and developers a huge head start.
- Embedded Java gives developers a wide choice of host OSes, from Linux to BSD to QNX to VxWorks and other RTOSes. With Android, you get a forked Linux kernel and no other options.
- Java is more flexible than Android, letting device developers customize and specify their own Java-based platforms without worrying about forking a rigid specification or implementation.

\(^2\) E.g., RTOSes like Nucleus, QNX and VxWorks
• Because Android tries to be a comprehensive applications platform, its device deployment footprint is quite large. By contrast, the run-time footprint of Java can be customized and scaled to meet a mix of application capability and bill-of-material requirements.

• Android and Java are both large open source projects. Despite an enthusiastic developer community, Android is largely captive to a single organization – Google. Compared to the Java Community Process, there is no real open process for introducing new functionality into Android, leaving OEMs and other developers to fend for themselves. In response, investments by OEMs and others in “differentiation” is rapidly leading to platform-level forking in the Android platform.

• Android is still fairly immature. To meet rapidly evolving requirements, Google is pushing out releases every 3-6 months. While “release early, release often” is the byword of open source software, with Android it is definitely too much of a good thing – OEMs, ISVs and other developers cannot keep pace with this frenetic platform release cycle, causing further, version-based fragmentation. By contrast, Java is mature and feature rich, with a well-established, developer-friendly release cycle, very well suited to the longer life cycles of fielded energy management and consumer electronics products.

Remote Administration and Device Management

Java-based devices across the Smart Grid must be manageable without hands-on intervention. A great many Smart Grid nodes are physically hard to reach, residing inside power plants, on power pylons, or embedded inside of other systems, appliances and construction. Also, many or most Smart Energy device types are by definition headless, without local displays or input methods.

OSGi

Given these constraints, a good candidate for Smart Grid device administration and management is OSGi\(^3\), a Java framework from the OSGi Alliance. OSGi comprises a dynamic module system for Java, with the OSGi Service Platform offering standardized primitives to support applications development from small, reusable and collaborative components.

OSGi provides functionality to update application composition dynamically across a variety of networks, without restarting managed devices by implementing a service-oriented architecture where components can dynamically discover one another. The OSGi Alliance has defined a range of standard component interfaces for common functions, including HTTP servers, configuration, logging, security, user administration and XML and many more. The OSGi Alliance model presumes that multiple vendors can supply standards-based plug-compatible implementations of these components, either as proprietary or open source software.

\(^3\) OSGi was formerly known as the Open Services Gateway initiative, now an obsolete designation
The OSGi Framework is divided into four layers.

- **Execution Environment** – basically, Java SE and standard profiles
- **Modules** – OSGi Modules define Java class loading policies that augment the Java model with private classes and customizable links among modules, fully integrated into the OSGi security model
- **Life Cycle Management** – adds dynamic installation and execution management of “bundles” of modules and resolution of dependencies among them
- **Services Registry** – a model for cooperation among bundles through shared objects and events

You can learn more about OSGi and available OSGi implementations, solutions and services at [http://www.osgi.org](http://www.osgi.org).

**Conclusion**

This white paper has examined key requirements for building and deploying Smart Grid devices into the rapidly evolving energy ecosystem.

In particular, it has highlighted how *smart energy* requires *smarter devices* – intelligent control systems that are easily provisioned, maintained and upgraded with both inherent functionality and after-market applications. At Oracle, we believe that embedded Java provides the best Smart Grid platform, with functionality for today’s legacy-centric grid, and scalability and performance for the emerging Smart Grid over the next decade.

**Smart Grid Resources for Java**

To complement this white paper, Oracle offers developers and integrators a range of horizontal resources for embedded Java, as well as programs and information specific to smart energy:

<table>
<thead>
<tr>
<th><strong>Java Platform Resources</strong></th>
<th><strong>Smart Grid Resources</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Java Developer Blog</td>
<td>Webcast: Java Products for the Smart Grid</td>
</tr>
<tr>
<td>Java Tech Network</td>
<td>Webcast: Smarter Devices with Oracle Embedded Java</td>
</tr>
<tr>
<td>Technical Embedded Java Product Information</td>
<td></td>
</tr>
<tr>
<td>Easy Cross Device UI Creation with LWUIT</td>
<td></td>
</tr>
<tr>
<td><a href="http://lwuit.java.net">http://lwuit.java.net</a></td>
<td></td>
</tr>
<tr>
<td>Java ME SDK Description</td>
<td></td>
</tr>
<tr>
<td>OSGi</td>
<td></td>
</tr>
<tr>
<td><a href="http://www.osgi.org">http://www.osgi.org</a></td>
<td></td>
</tr>
</tbody>
</table>
Making the Smart Grid Smarter with Embedded Java
March 2011
Author: Bill Weinberg, LP-Analytics

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Copyright © 2011, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. UNIX is a registered trademark licensed through X/Open Company, Ltd. 1010

Hardware and Software, Engineered to Work Together