Unified Device Management

via Java-enabled Network Devices

Tal Lavian
Rob Duncan
Agenda

• Unified Management for Unified Networks
• Openness - Virtual community development, Domain experts
• Open Service Interface - values
• Architecture and technology concepts
• Strong security
• Java SNMP MIB API
• Summary
Purpose

• To introduce the new Open Networking Architecture that is based on Java-enabled Network Devices
Unified Management

OBJECTIVE
Unified management

SOLUTION
Java “Optlets” on all devices
Security and Directory

BENEFITS
Java-Enabled Network Devices

Java on all devices
Unique value of Java

Unified Management
Community Openness

• Success stories by large community of developers

• Net-Based developers’ communities
  – Linux, GNU, Apache, BSD, X-Windows, Perl, Tk/Tcl
  – Netscape browser, NFS, JDK, JVM

• Linux everywhere:
  – Compaq, HP, IBM, SUN and SGI.
  – Intel, Sparc64, Alpha, PowerPC

• The Web Changes everything
  – Java, XML, E-Business
Open Service Interface - Value Propositions

• An open device software architecture enabler that:
  
  – *Reduces development cost* by enabling cross-platform development.
  – *Improves TTM* through “feature-on-demand” capabilities.
  – *Increases product differentiation* by allowing incremental customization of products.
An open device software architecture enabler that:

- Enhances scalability and flexibility for distributed deployment of management and IP services.
- Facilitates innovation by opening devices to third party developers.
- Provides incremental revenue through potential consulting/ customization services.
Open Service Interface - Levels of Adoption

Device Level Enabling Technology

- Selected device implementation.
- Feature-on-demand capability.
- Development efficiency.

Open Systems Architecture

- Opening up of APIs for:-
  - Customers.
  - Consulting services.
  - 3rd party ISVs.

Distributed Applications Framework

- Common distributed features.
- Distributed NMS applications “Optlets”.
- Mobile Agents

Phase 3

Phase 2

Phase 1

Phase 4

New Type of Applications

- Innovation
- Imagination

Value

Time
Java-enabled Network Devices

• What we have accomplished:
  – Java-enabled Device Architecture
  – JVM for Switch, Router
  – JVM for Network device
  – Java SNMP MIB API
    • include proxy mode for devices with no JVM,
Technology Concept
“Reversed Applets”

Web Server → Applet → Web Browser

The JVM is in the Browser

Technology is based on the concept of Reversed-Applets

Web Browser → NMS

Java Beans → Optlets → Reversed Applets

The JVM is in the Device
Potential Applications

• “Feature-on-demand” for devices
• New class of system level Optivity applications in the form of distributed “Optlets”
  – Characterized by system applications that require intensive interaction between NMS and device and/or across multiple devices.
  – Potential applications are topology, design analysis, diagnostics, policy implementations.
Benefits and Value

• Enabling component of a new intelligent network architecture
  – Distributed applications-on-demand.
  – Component of AI (artificial intelligence) enabling infrastructure.
  – Roaming diagnostics and self-healing capabilities.
  – Built-in support for open industry ISV support.
Example - Local Intelligence

Download

Security

On-demand

Authentication

Intelligence

Monitor

React

JVM

OS

HW

Network Device
Application Example

- Download Intelligent Agent Monitor from NMS to the device.
- Wait for threshold.
  - Might be complex conditions
- Send “condition exceeded” event to NMS.
- Automatic download appropriate application
- Application takes action.
System Architecture

“Optlets” = Distributed Optivity Applications
JSCP System diagram

Java Applications

JVM

Java Libraries

Services

Reset
Clean-up
Configuration
Event Mapping
Monitor Thread

JNI

Memory Manager

Scheduler

Native Applications

RTOS & Hardware

Context Switch
Strong Security in the new model

• The new concept is secure to add 3rd party code to Nortel devices
  – Digital Signature
  – Nortel “Certified Optlet”
  – No access out of the JVM space
  – No pointers to harm the work
  – Access only to the published API
  – Verifier - only correct code can be loaded
  – Class loader access list
    • Different Optles with different access levels
  – JVM has run time bounds, type, and executing checking
Old model Security (C/C++)

- Old model - Not secure to add 3rd party code
  - Not recommended to add 3rd party code
  - Dangerous, C/C++ Pointers
    - Can touch sensitive memory location
  - Risk: Memory allocations and free
    - Allocation without freeing
    - Free without allocation (core dump !!!! )

- Limited security in SNMP
Java SNMP MIB API

• Portable across a range of network devices
• Extensible
• Simple and convenient for client use
• Consistent with SNMP model
• Hide unnecessary SNMP details
• Permit optimized access
• Re-use MIB documentation
MIB API Generation

• Most of the Java code is generated automatically
• ASN.1 MIB definitions are converted into Java classes
• Documentation and commentary in the MIB definitions is placed as Javadoc formal comments
• HTML documentation generated from Javadoc
MIB Objects

• The MIB data model is structured as a tree
• API represents MIB groups with Java classes
• MIB variables are represented with accessor methods
• Conceptual tables are represented with iterators
• API converts SNMP data values into standard Java types
**JSNMP MIB API Architecture**

- API uses a MIB Map to dispatch requests to variable access routines
- Different parts of the MIB tree can be serviced by different mechanisms
- Two main schemes:
  - An ad hoc interface to the SNMP instrumentation layer
  - A generic SNMP loopback
Advantages of MIB map

- Allows immediate generic implementation of the entire MIB via the loopback scheme
- Enables optimized native implementation of key MIB variables for maximum efficiency
- Permits definition of pseudo-MIB variables for extending MIB dynamically
- Provides site for centralized access management
Java MIB API - Proxy mode

- Uses SNMP loopback mechanism to target a remote network element
- API can be used to control devices that don’t have an embedded JVM
Summary

- Openness - successfully proven paradigm
- Domain experts - virtual community
- Allows innovations and added value

- *dynamic* agents vs. *static* agents
- Dynamic Loading
- Strong Security
- An enabling-technology

- Take it, and make it work for you