Java Management APIs

Tom LaStrange

tom.lastrange@central.sun.com

Java Management APIs (JMAPI) Are:

A collection of Java classes, that extends the Java Foundation Classes, to allow rapid development of integrated management solutions. Management covers heterogenous systems, networks and services that comprise the enterprise computing environment.

Why Java?

- **Eliminates the agent distribution problem**
  - Only need the Java runtime on the target device. Ship the operations on demand.
- **Eliminates the versioning problem**
  - Since we ship the operation, there is less need to worry about protocol or interface compatibilities.
- **Extensibility**
  - Operations can be dynamic and shipped to the target device.
- **Heterogeneity**
  - Solaris, Win95, other Unixes are already supported. Goodness for both UI and framework.
- **Leverage**
  - Many problems (e.g. security) are being worked across the industry. It's the language of the Internet.
- **Security**
  - Java provides a much better model to implement real security.

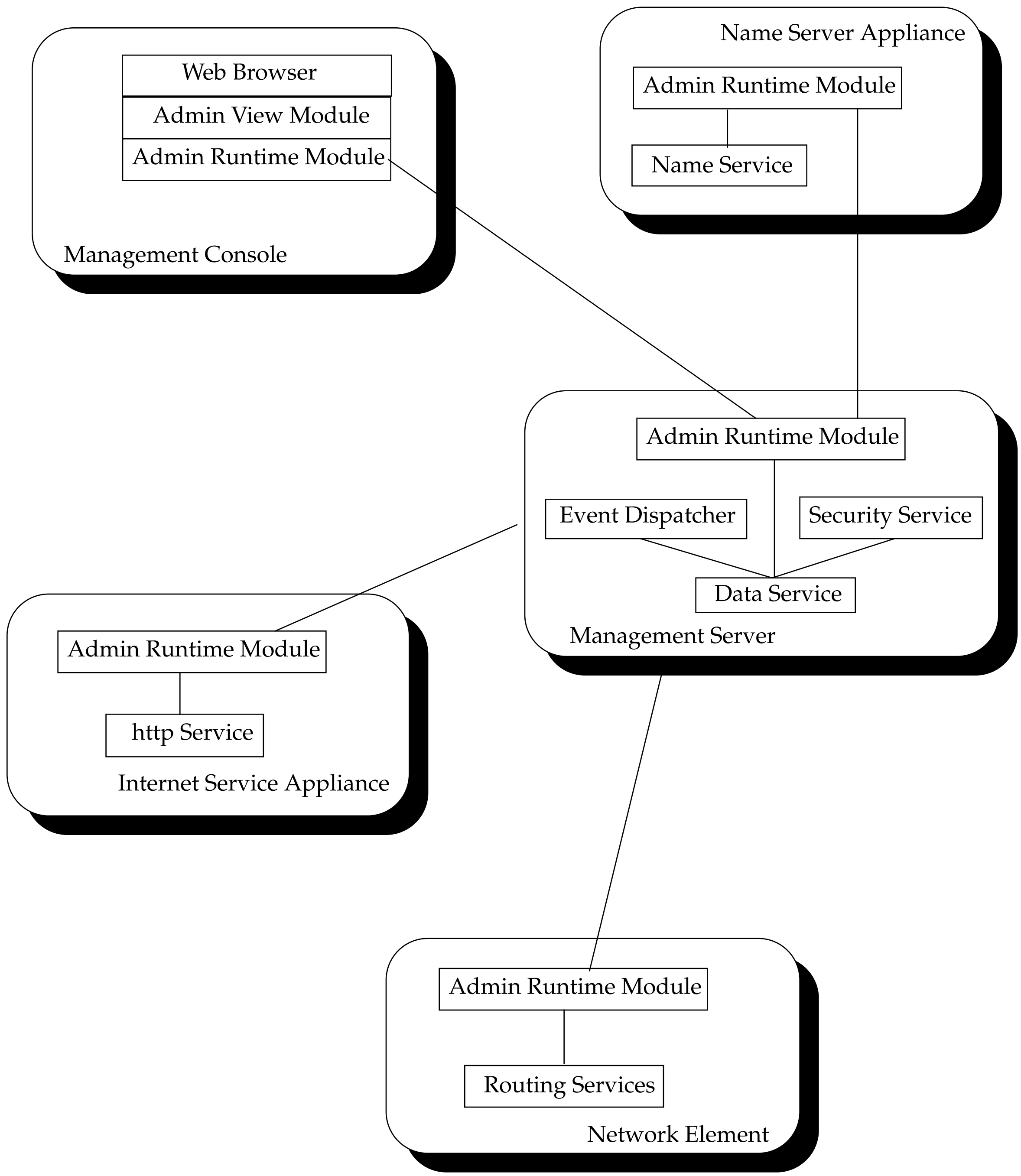
High-Level Architecture

- **Distributed Management Console**
  - Where the administrator issues operations.
  - Supports a web browser and standalone applications.
  - Applications can have a GUI or CLI.

- **Distributed Management Server**
  - Provides the key management services of security, event dispatching and data access.

- **Appliances**
  - Managed machines.
  - Ideally, these require little or no management.
  - Can assume many different roles (e.g., desktops, name service servers).

Possible Configurations



JMAPI Components

- User Model
- Admin View Module
- Admin Runtime Module
- Admin Services
- Base Objects

User Model

- **Object oriented interaction model**
  - Properties and relationships are ubiquitous
  - Application boundaries are seamless
- **Browser-based navigation model**
  - Reduces window management
  - Encourages exploration
- **Interactive assistance for complex tasks**
  - Supplement to primary interface
  - Establish guiding task oriented dialog w/ user
- **Extensible components**
  - Management pages
  - Containers
  - Property books
  - Task managers

Why a Web Browser?

- **Heterogeneity**
  - Web browsers give us the capability to run the same "application" on different platforms. Eliminates duplicate efforts.
- **Good User Interface Model**
  - Administrators spend most of their time doing information process. The hypertext model maps extremely well onto our problem space.
- **Market Demand**
  - Over 10 million users spend a significant amount of their computing time using a web browser.
- **Leverage**
  - Information pre-processing and role-up technologies are evolving quickly to keep up with the demand for such capabilities.
- **Extensibility**
  - The web browser model can be extended by modifying applets/CGIs and using the database to track these extensions.

Management Pages

- **Definition:**
  - Pages are the primary mechanism for accessing status, relationship, and configuration information about managed objects. They are composed of one or more containers. Navigation within the user interface is primarily page-to-page.

- **Types:**
  - Pre-defined and custom
  - Container-based and property books

- **Notable Characteristics:**
  - Additional pre-defined pages can be installed
  - Users can create and save custom pages
  - Turns window management problem inside out.

Containers

- **Definition:**
  - Containers collect and display managed objects. Each container displayed on a page maintains its own selection methods and page, view, filter, and sort properties.
- **Types:**
  - Predicate-based and ad hoc
  - Hierarchy
  - Topology
  - etc...
- **Notable Characteristics:**
  - The set of container types is extensible
  - One or more per management page
  - Are selectable and can be configured (e.g., filter, sort, display objects as graphs, etc).

Property Books

- **Definition:**
  - Property books are the mechanisms by which users configure new or existing objects.
- **Subtypes:**
  - none
- **Notable Characteristics:**
  - Are extensible through new configuration modules
  - Are dynamically constructed from index file in database
  - Modules register with book level controls

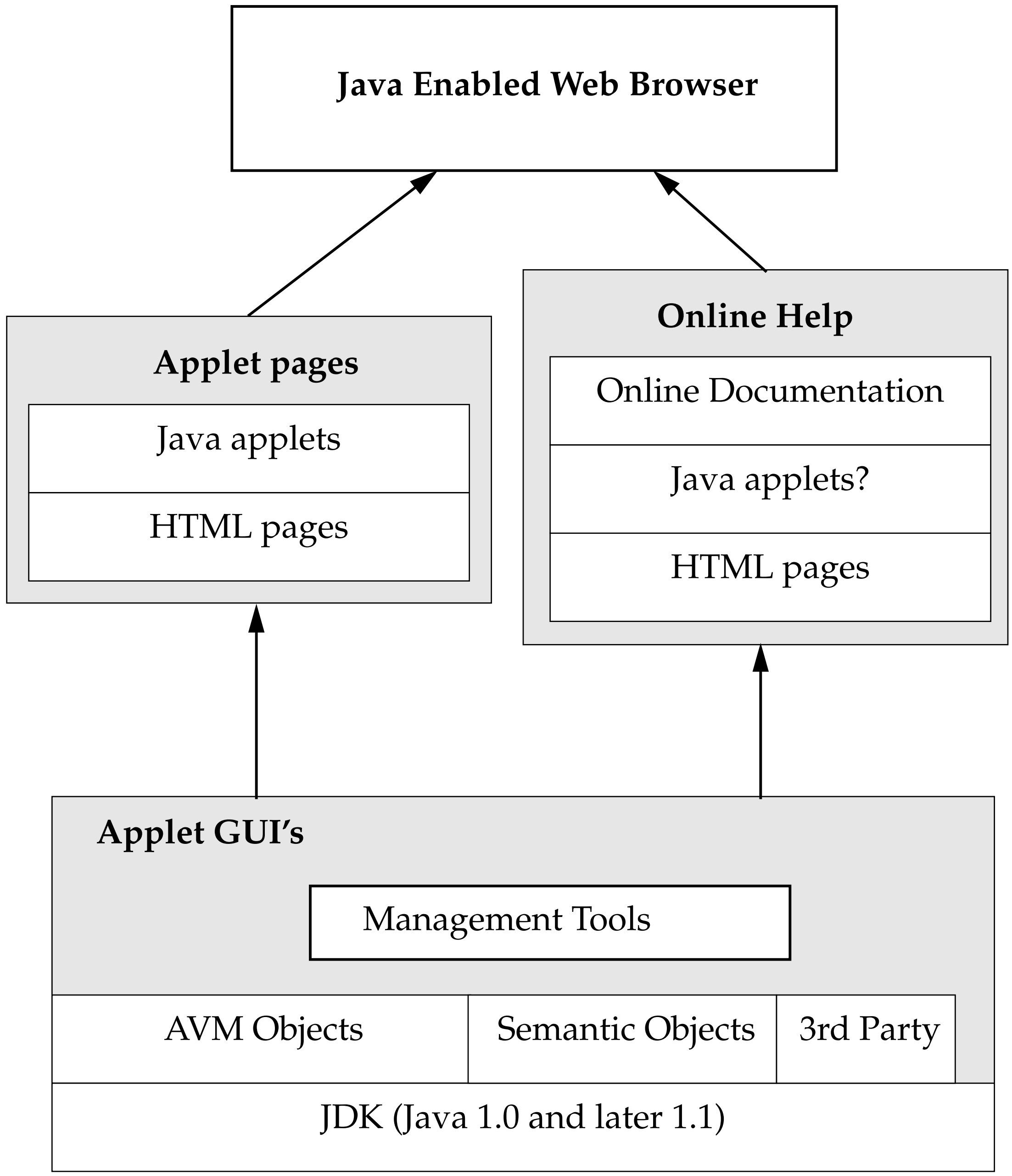
Task Managers

- **Definition:**
  - A task pages helps users achieve a desired end state that would normally require completion of a complex or large set of procedural steps. It does so by establishing a guiding task oriented dialog with the user that hides many of the steps and/or much of the complexity that would be involved in performing the tasks directly with the applications primary interface.
- **Subtypes:**
  - none
- **Notable Characteristics:**
  - New Task Managers can be added and linked into appropriate help systems, object menus, and global index
  - Hide steps and reduce complexity
  - Often cut across object and service boundaries

Admin View Module

- **Provide common UI components**
- **Consistency**
  - Promote consistent style for Internet management applications.
- **Extensible**
  - Build upon existing Java objects.
  - Architect so objects are usable by other groups.
  - Reuse, objects shared among projects.
- **Efficient**
  - Architect for small memory footprint objects.
  - Cache common images.
- **Support User Interaction Model**
- **Don't reinvent the wheel**
  - Port existing Motif code where appropriate.
  - Use 3rd party tools and class libraries.
  - Look at public domain examples.

AVM Architecture



AVM Classes

- **Image Button**
  - 3D Button Containing an image
- **State Button**
  - An Image Button that toggles pixmaps when activated.
- **Scrolling Window**
  - A general set of classes that will scroll any component
- **Hierarchy Browser**
  - A hierarchy browser with collapsible nodes
- **Property Book**
  - A dialog consisting of a property browser on the left hand side and property pages on the right hand side
- **Table**
  - Multi-column lists with glyphs
- **Convenience Dialogs**
  - Like error, save and information dialogs in Motif

AVM Classes

- **Pop-up Menus**
  - Java group supplying in Java 1.1
- **Toolbar**
  - Basic Toolbar like in Windows
- **Page Header/Title Bar**
  - Title bar for pages containing a gif and a textual title
- **Help System**
  - Simple html-based help system
- **Link Label**
  - Label that acts like a link
- **Image Canvas**
  - Wrap an image in a Canvas so it can be displayed/scrolled
- **View Configuration**
  - Views which contain lists of items that may be configured in a number of different ways

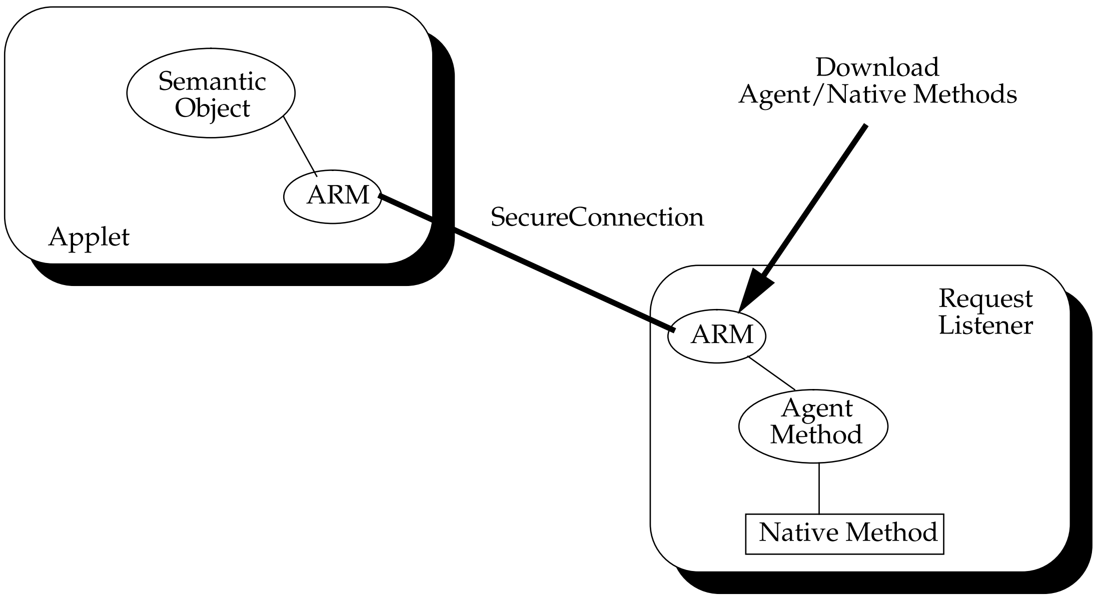
Application Integration

- **No more applications!**
- **Need to think in terms of link management pages.**
- **All "class browsers" are visible through the console home page.**
- **Console can be customized to support any number of different views.**
- **Registration Tool**
  - Registers Applets
  - Registers Management Pages
  - Register links to other objects and their management pages
  - For new objects we need class, instance and containment criteria
  - On-line help

Admin Runtime Module

- **Secure, Distributed Management Operations**
- **Eliminates Agent Versioning and Distribution Problems**
- **Model for Representing Distributed Resources**
- **Heterogeneity**
- **Extensibility**
  - Additional attributes
  - Execution of programs
- **Protocol Independence**
  - Will provide support for communicating with SNMP agents.
  - Recognize the legacy issues.

ARM Architecture



ARM Components

- **Semantic Objects**
  - Represents distributed resources and services to the application developer
- **Agent Methods**
  - Downloaded on the target system to perform a specific piece of management functionality
- **Native Methods**
  - Platform specific management operations
- **SecureConnection**
  - Permits multiple protocols
  - Public-Key based security for remote message passing
- **Secure Class Loader**
  - Securely loads classes over the network
- **Java Virtual Machine**
  - Standard Java runtime and objects

Admin Services

- Data Service
- Security Service
- Containment Service
- Notification Service

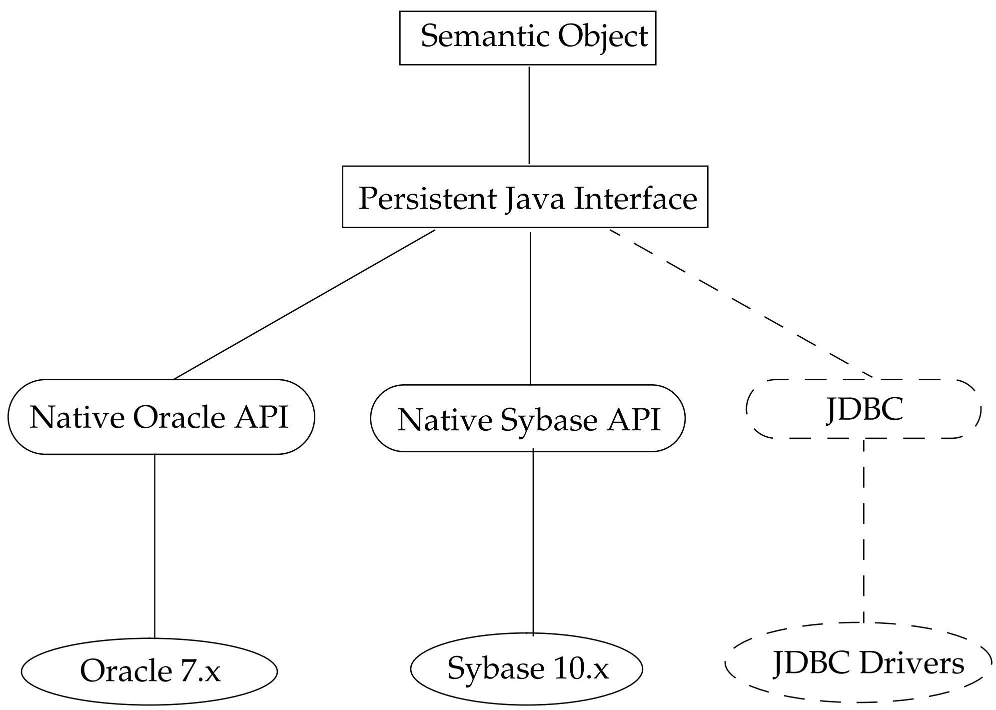
Data Service

- **Real Database Persistent Storage**
  - Supports multiple database engines
  - Will have a low-end database engine
- **Canonical Form all classes**
  - Register classes as part of installation
- **"Golden" copy of management data**
- **Backing store for other services**

Why a Database?

- **Required Capabilities**
  - Files and name services can't give us the availability and reliability of administration data.
- **Heterogeneity**
  - Can contain a single, logical view of all the administration across the enterprise.
  - Can define resources that map onto something other than /etc file.
- **Scalability**
  - When workgroups share information the database can keep information consistent and can store and retrieve large amounts of information.
- **Extensibility**
  - Allow a place to store and eventually share users extensions.
- **Leverage**
  - Policy, and other management solutions can all leverage the database capabilities.

Data Service Architecture



Security Services

- Working with JavaSoft and will build on what they provide.
- Browser class loading is currently focused on safety and not an overall security model
- Signed Applets
- Distributed authentication
- Encrypted Channels
- Breaking a local hosts root WILL not result in breaking network security.

Containment Service

- **Basic grouping mechanism**
- **Necessary for scaling**
- **Objects may belong in multiple containment relationships**
- **Containments are named**
- **Two styles of containment construction**
  - User Extended
  - Predicate-Based

Notification Service

- **Simple Producer/Consumer Model**
  - Avoids Polling
  - Asynchronous Notifications
- **Notification primary functions are registration and forwarding**
- **Filtering is specified by either Producer or Consumer.**
- **Logging is done only on the location of an instance of a notification service.**
- **Goals are simple and light-weight**

Basic Objects

- **Each Object will have:**
  - Property Books
  - Class Browser
  - Simple Discovery
- **Management Interfaces Only**
- **Enough for Integration. Not all things to all people.**
- **Current List**
  - System
  - Person/User
  - Network Element
  - File System Service
  - Storage Device

Status

- **Announced at JavaOne May 29, 1996.**
  - Focus is on AVM classes for demo
- **Joint Demo/Announcement at PC Expo June 18, 1996. Partners included:**
  - Cisco
  - Legato
  - JavaSoft
  - Platinum
  - 3Com
  - Central Design Systems
- **Incremental Releases**
  - No Big-Bang FCS; Doesn't work because of "Internet Dog Years"
- **AVM and ARM classes are almost complete**
- **Early access in the fall, complete by the end of the year. JMAPI will be free over the net.**
- **Demo can be seen and downloaded from**

http://java.sun.com/APIext/JMAPI/