Building Installers for the Java Platform: the Results of JSR 38

Paul Wolgast, Jet Propulsion Laboratory
Paul Lovvik, Sun Microsystems Inc.
Brad Andersen, Tripwire Inc.
Eric N. Shapiro, Zero G Software Inc.
Presentation Goal

- To inform you about the ongoing specification work for cross platform installer and builder API specification, Java Specification Request (JSR) 38
Objectives

• As a result of this presentation, you will:
  – better understand the installer architecture
  – learn how to build your own installer
  – learn where to download the demo products
  – learn how to get access to the specification
  – learn how to get involved in the Java Community Process (JCP)
Presentation Topics

- JSR 38 runtime features and architecture
- GUI installer builder and demonstration
- Cross platform installer and demonstration
Problems Addressed by JSR 38

- Cross-platform application deployment is the rule, not the exception
- Handle inter-application dependencies
- Manage conflict resolution
- Sufficient flexibility to address custom install problems
- Open solutions generally unavailable
JSR 38’s Approach to the Problem

- Provide level of abstraction between O/S and services
- Provide a product registry
- Provide extensibility at service and UI layers
- Create a public specification and open implementation through the Java Community Process
What JSR 38 is Addressing

- Builder APIs
- Compatibility Test Suite
- Installer Runtime Framework
Commercial Origins

- Originally based upon Sun’s WebStart 3.0 product with these new features:
  - eliminates dependency on a custom class loader
  - self extracting jar file as a distribution vehicle
  - accommodates remote installation
  - extensibility designed up front
  - accommodates Swing, AWT, and console user interfaces
  - open source, will be freely available
The Other Web Start

- How does JNLP (JSR 56) relate to JSR 38?
  - web centric model, no installation phase
  - describes appl. package on a web server
  - standard execution environment for appl.
  - designed for an online strategy

- JNLP is "not a general installer for applications"

- JNLP could launch J|FI jar files

- JNLP could install JRE etc.
Runtime Features

- Cross platform
- Extensible, no redelivery
- Self contained executable
- Multi-language
- Remote execution
- UI agnostic
- Service rich infrastructure
- Product registry

Swing
Awt
Tty
Runtime Architecture: Lifecycle

Executable Installer Archive

Welcome to the installation of xyz
Press the Next button to proceed.

<Back  Next>  Cancel

Launcher

Installed Bits

Registry
Runtime Architecture: Communication Bus

- Normalizes communication between services
- Facilitates transparent remote operations
- Provides coordination and synchronization of services
- Provides execution of service methods anywhere in the system
- Allows for future growth of integrated services
Runtime Architecture: Standalone Operation

Card 1  Card 2

Proxy 1  Proxy 2

application object
Runtime Architecture: Remote Operation

Diagram showing the architecture with Local JVM, Remote JVM, Card 1, Card 2, Card 3, Proxy 1, Proxy 2, Proxy 3, RM, and Bus.
Runtime Architecture: Services

- UI/navigation
- File
- Filesystem
- Archive
- Product definition
- Product registry
- Logging
UI Service Architecture

UI Controller
pageDefinition
setDelegate(UIGroup)

Set of pages and navigation rules

UIGroup Selection

AwtUIGroup
SwingUIGroup
TTYUIGroup
SilentUIGroup
UI Service Architecture: Navigation

- UI Navigation Model
  - Page definition members
  - Iterator
  - Page definition
Pages and Navigation: User’s View
Pages and Navigation: Page Definition

- Top Folder
- Navigator
- Page 1
- Page 2
- Folder
- Display Rule
- Page 3
Pages and Navigation: Page Display
Platform Abstraction Overview

File
Windows Solaris Java

Product Registry
Windows Solaris Java

Desktop Services
Windows Solaris Linux

Registry

Operating System
File Service Architecture

- setDelegate(service)

File Service Selection

- Pure Java 1.1
- Pure Java 1.2
- Solaris
- Linux
- Windows
- ...

JavaOne
File Service Architecture

- File service selection criteria:
  - Can the service implementation be instantiated?
  - Is the runtime platform acceptable to the service?
  - Are the required resources available?
  - Select the remaining service with the most methods
File Service Architecture

Simple File
Java 1.1 File Interface

File Attributes
get/set Attributes

File Owner
get/set Owner

Simple File 2
Java 1.2 File Interface

File Group
get/set Group

Linux File Bean
Proxy that implements interfaces appropriate for Linux

Linux File Service
Native Impl. for Linux
SimpleFile newFile(String path)
File Service Architecture

Local JVM

Bus

File Service

Service Delegate

Linux File Service

→ setOwner();

Remote JVM

Bus

Linux File Bean

Client

→ setOwner();

SimpleFile file =
FileService.newFile(filePath);
if (file instanceof FileOwner) {
    ((FileOwner) file).setOwner(ownerID);
}
Product Definition Service

- Product
- Feature
- Component
- Installable units
  - not registerable
  - extensible
  - current support for:
    - platform independent file
Product Definition Service

- Installable Units (cont)
  - future support for
    - zip file
    - RPM
    - Solaris Package
    - tar file
    - HP swinstall
    - AIX installp
    - CAB file
Registry Service

- Windows Registry interface
- Product Installation Registry
  - pure Java implementation
  - preserves the user’s view of installed components
  - provides for dependency checking and enforcement
  - creates single point of access for uninstall
  - facilitates smarter installers and uninstallers
Registry Service
Registry Service

[Image showing a software interface for Solaris Product Registry with details about SUNWtools.

Title: SUNWtools
Version: 3.0.1 2/02
Location: /opt
Client Size: [details]
Installed On: [details]
Installed From: Webstart Wizard SDK 3.0.1
Vendor: Sun Microsystems, Inc.

More Information:
Unique Name: SUNWtools.
Supported Languages: en
Component Type: Component
Dependent Component(s):
  Linux Samples
pkg: SUNWtools
source: Webstart Wizard SDK 3.0.1]
Archive Service

- Provides access to all classes and files in the archive
- Allows us to support multiple formats:
  - jar
  - zip
  - other proprietary formats
Building Installers

- Java Program
- Installer SDK
- Product Definition
- Page Definition
- Installer Runtime
- Executable Installer Archive
Demonstration of GUI Builder

- Demonstration of GUI Builder
Results of Builder Execution

- Executable installer archive
  - can be run standalone
  - can be run over the network

- Archive contains everything:
  - the launcher
  - the installed bits
  - the product definition
  - all required runtime classes
Standalone Archive Execution

Executable Installer Archive

Welcome to the Installation of xyz
Press the Next button to proceed.

Launcher

Installed Bits

Registry
Remote Archive Execution

Executable Installer Archive → UI Launcher

Welcome to the Installation of xyz
Press the Next button to proceed.

<Back  Next>  Cancel

(UI Runtime) → Local JVM

Remote JVM

System Launcher → System Runtime

Installed Bits → Registry
How to Get Involved

- Join the JCP
- Download the sample binary material
- Provide feedback directly to JSR 38 at: comments@jsr38.jpl.nasa.gov
The Java Community Process

- The JCP is dependent on the Java community for support
- Information on JCP membership: http://java.sun.com/aboutJava/communityprocess
- Draft specification available for comment by JCP members
- Final specification available for comment by general public
Downloads

- For sample binary material:

- To get the Web Start Wizards 3.0 SDK:
  - http://www.sun.com/solaris/webstart
Summary

- Architecture provides open, extensible installation solution
- Addresses problems:
  - cross-platform application deployment
  - inter-application dependencies and conflict resolution
- Will provide flexibility to address custom install problems
- The JCP is the means to participate
Q&A
JavaOne™
Sun's 2001 Worldwide Java Developer Conference®