ORACLE

Facilitating "Application Specific" aka "Stripped" Implementations

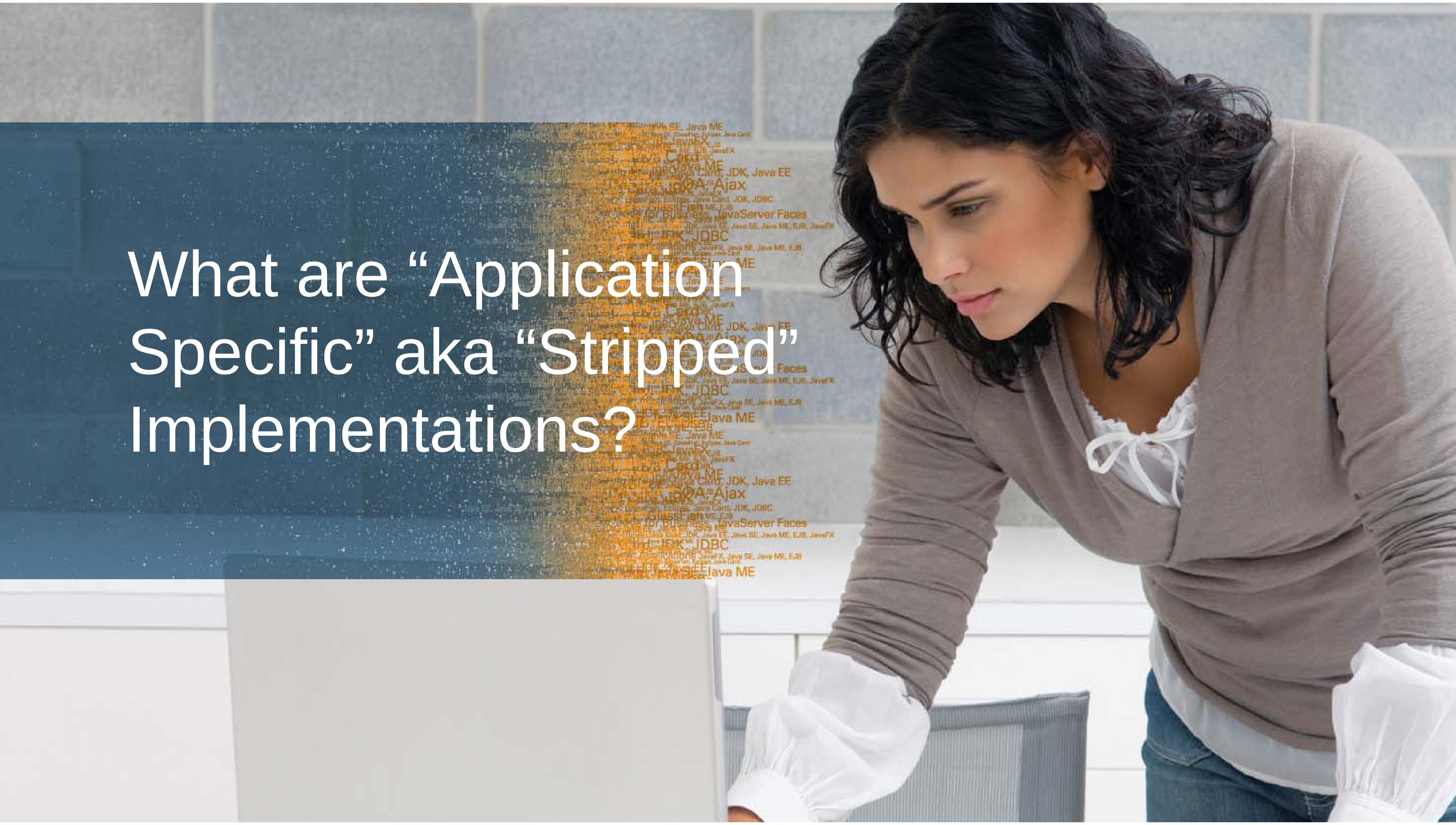
JCP EC Discussion
August 12, 2014

DRAFT FOR DISCUSSION v4

DRAFT – FOR DISCUSSION PURPOSES

Goal:

- Allow unused elements (e.g., methods, classes or even whole packages) to be removed or ‘stripped’ from a TCK-compliant implementation (e.g., Java SE and Java ME, but other specifications if desired), to reduce storage and memory consumption.



What are “Application Specific” aka “Stripped” Implementations?

"An implementation based upon a complete and TCK-compliant (e.g., Java SE or Java ME) implementation, but distributed with a dependent application that uses the implementation in a closed environment where unused elements are removed, or "*stripped*", in order to reduce storage and memory consumption.

DRAFT – FOR DISCUSSION PURPOSES

Application Specific Implementation - Basics

AKA “Stripped Implementation”

- Based on complete and TCK Spec compliant implementation
- Distributed only with a Dependent Application
- Unused elements may be removed, or ‘stripped’ to reduce storage and memory consumption
  - E.g., methods, classes or even whole packages
  - Manually, via provided tools, automated on deployment, etc.

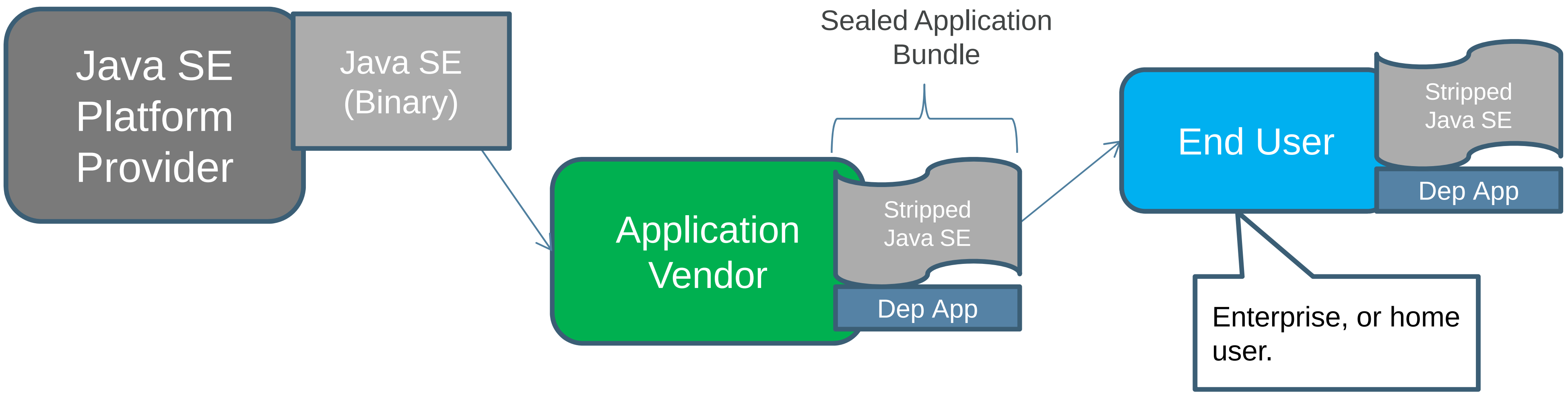
What Application Specific Implementations are *NOT*

AKA “Stripped Implementation”

- Not Compact Profiles
  - Compact Profiles are pre-determined spec defined subsets. Stripped Implementations are unique and specific to an application(s)
- Not Modularity
  - Modularity is a proposed future feature with spec defined implementation patterns and related modularity framework. Stripped Implementations would enable immediate and ad hoc stripping.
- Not “Stripped Platforms”
  - Goal is to enable stripping for specific Applications, not to define new stripped platforms, which would lead to fragmentation.

Application Specific Implementation

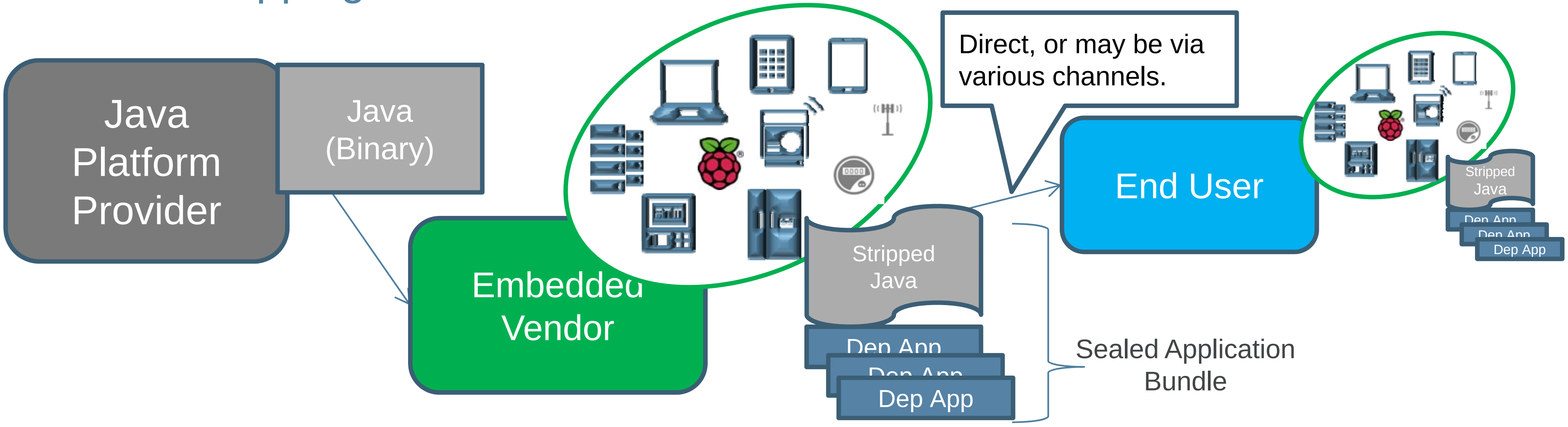
Ex 1: ‘Stripping’ and Redistribution of Java SE by Application Developers



- An Application Developer licenses Java SE from a Platform Provider, “strips” it with their dependent application, creating a “Sealed Application Bundle” and redistributes it further
- Either Java SE or Java ME (perhaps other JSRs)

Application Specific Implementation - Embedded

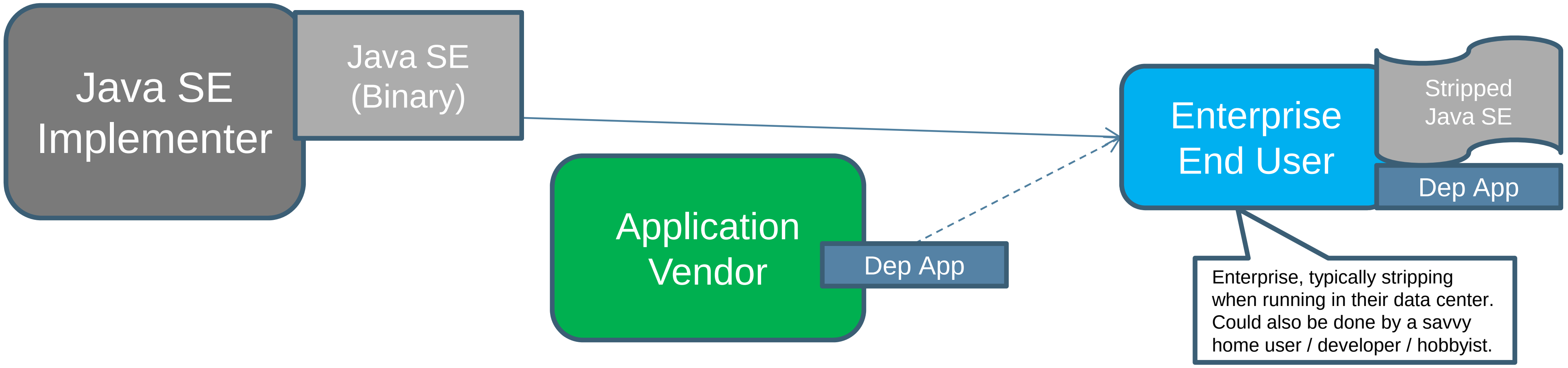
Ex 1a: ‘Stripping’ and Redistribution of Java SE in Embedded Scenario



- An Embedded Vendor licenses Java from Platform Provider, “strips” it with their dependent application(s), creating a “Sealed Application Bundle” and redistributes it further on hardware
- Either Java SE, Embedded or Java ME (perhaps other JSRs)

Application Specific Implementation

Ex 2: ‘Stripping’ of Java SE by an Enterprise User


Java SE Implementer
Java SE (Binary)
Application Vendor
Dep App
Enterprise End User
Stripped Java SE
Dep App
Enterprise, typically stripping when running in their data center. Could also be done by a savvy home user / developer / hobbyist.

- An End User either builds their own dependent application, or licenses one from an Application Vendor, and then ‘strips’ an implementation provided by a Java SE Implementer
- Either Java SE or Java ME (perhaps other JSRs)

Sounds great,
ship it!

Additional Constraints

Protecting Fragmentation and Compatibility

Application Specific Implementations must:

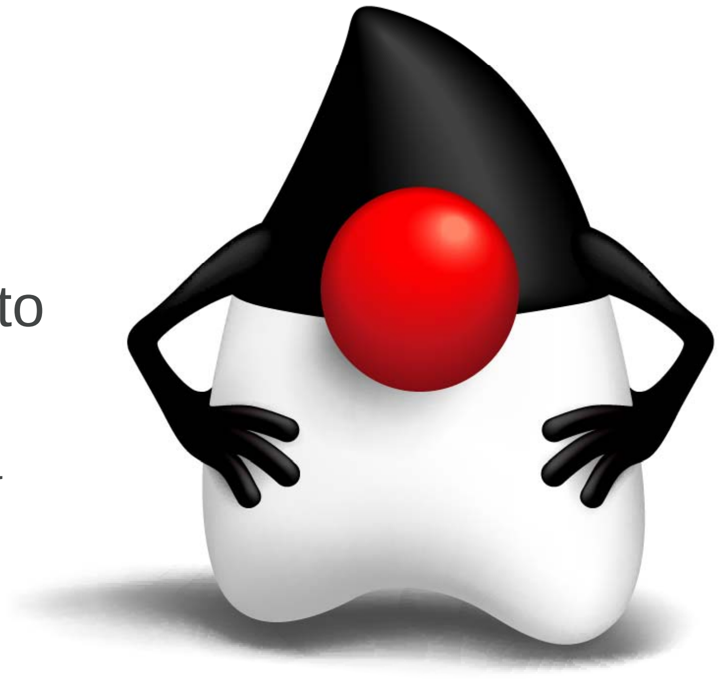
- Function identically to the ‘non-stripped’ Full Implementation
- Be closed once stripped – no in or out of new functionality or code:
  - Be restricted from further stripping or other modifications to the app downstream once created
  - Do not expose APIs and cannot execute code other than the dependent application(s)
    - To prevent fragmentation of platform. Application developers should always start from Full Implementation.

Requires changes to Licensing and Specs

DRAFT – FOR DISCUSSION PURPOSES

Licensing Proposal

- Make completely optional for Platform Providers (Spec Implementers) to allow “Stripping” of their implementations
- Require the “Stripper” to enter agreement with Spec Lead, and pass a TCK specific to Stripping
  - Application Developer, End User or even a Java Implementer
- Create an enforceable relationship with Spec Lead

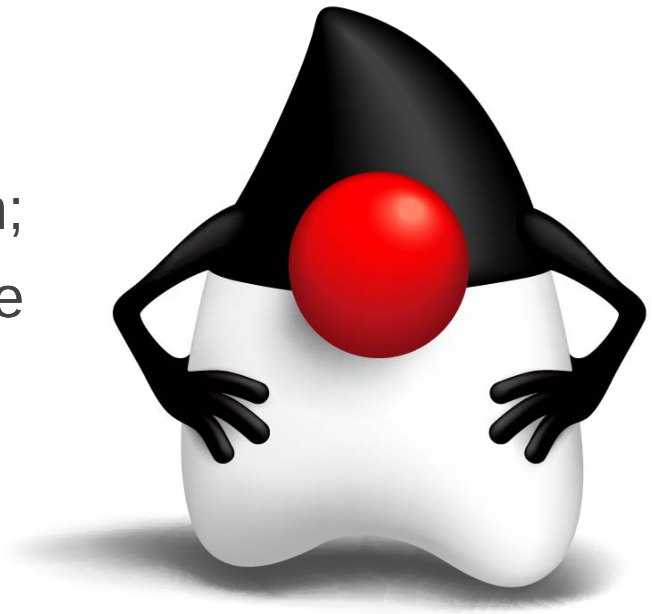


DRAFT – FOR DISCUSSION PURPOSES

Optional Part of TCK

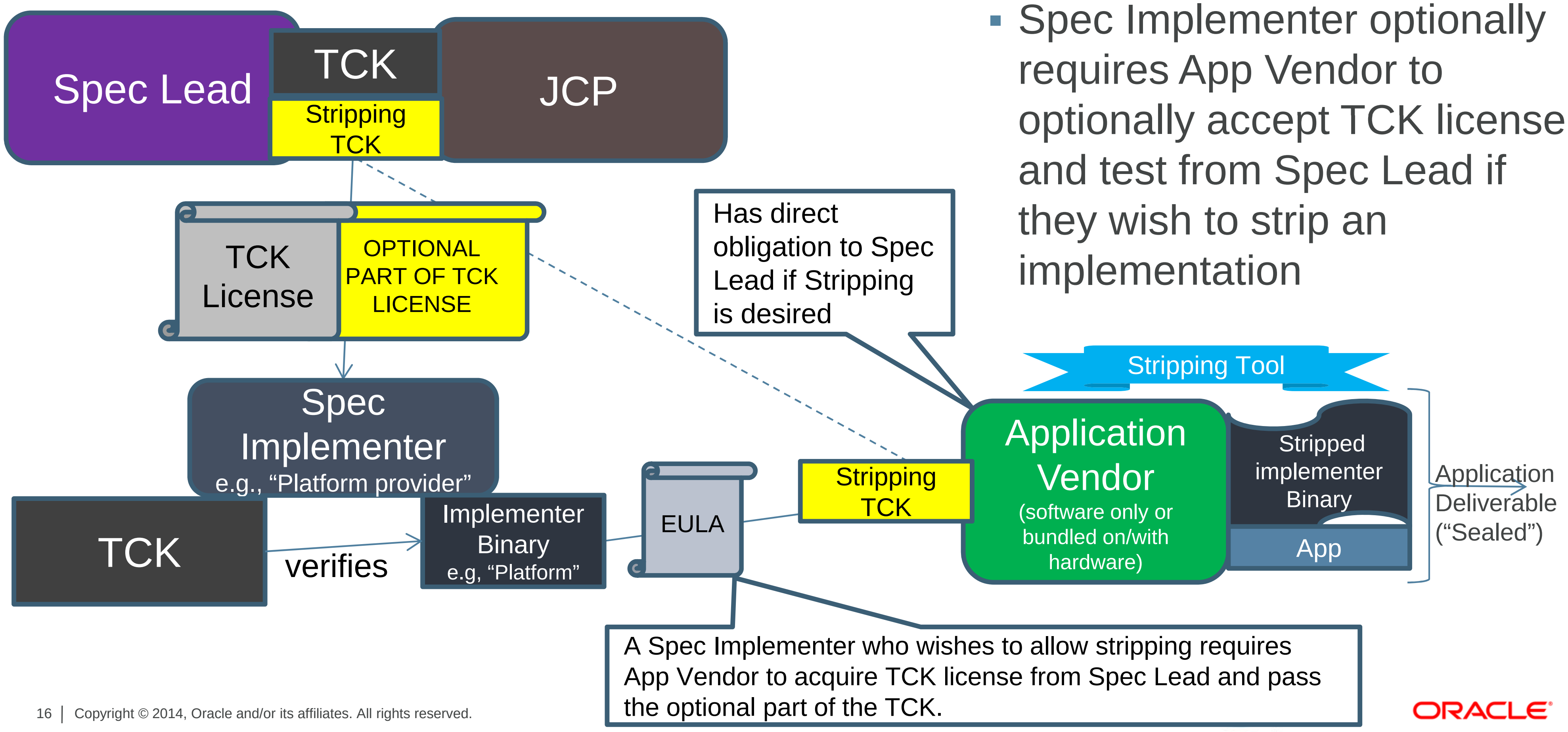
Example "tests" in the optional TCK specific to Stripping

- Your stripped implementation is:
  - Derived from a complete, conventionally compatible implementation of the platform;
  - Does not expose APIs and cannot execute code other than the included Application;
  - Functions identically to how it functions with the Full implementation.
- May just be a 'checklist' vs provided software test suite



DRAFT – FOR DISCUSSION PURPOSES

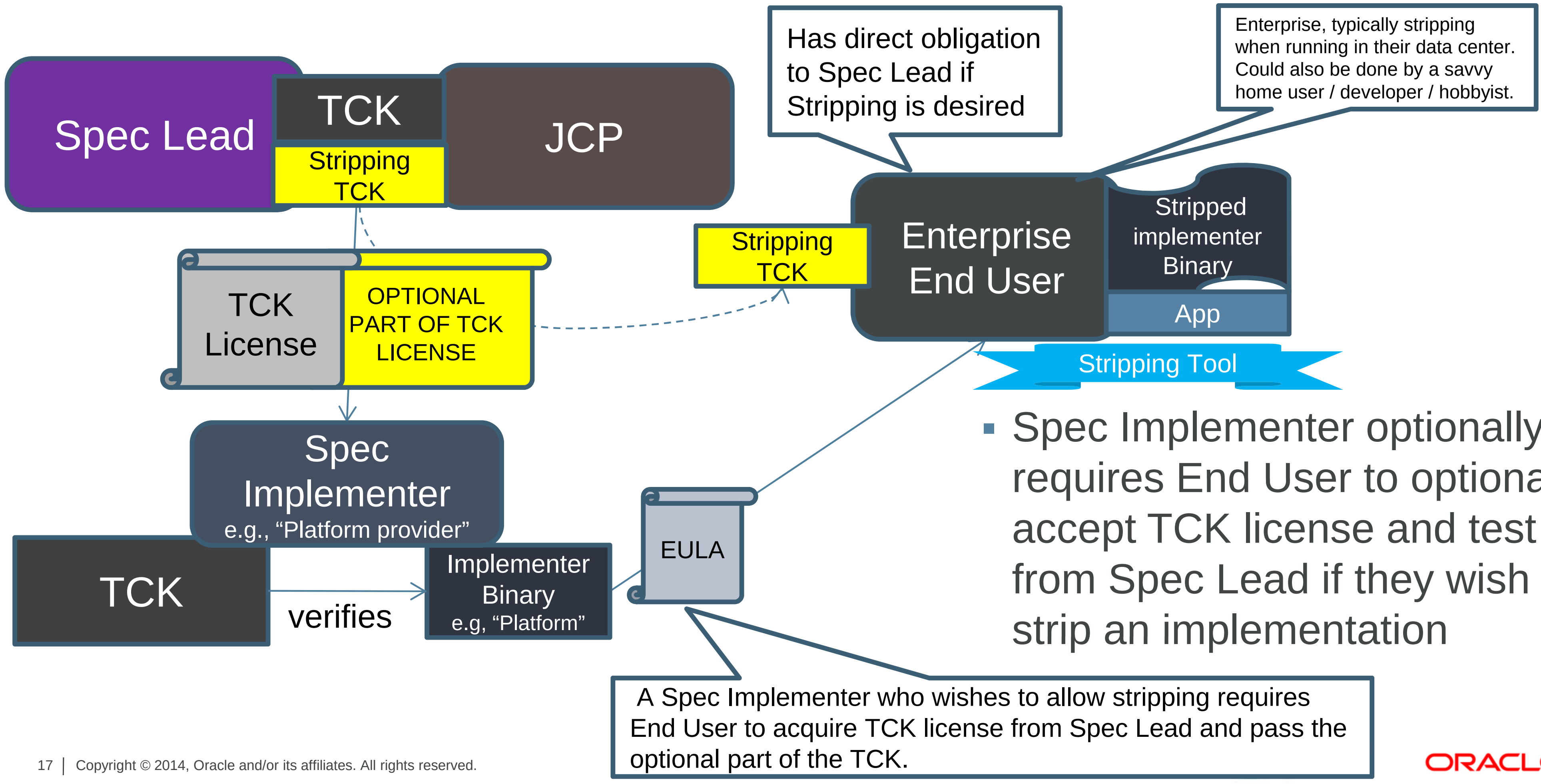
Licensing Proposal – App Vendor POV


Spec Lead
TCK
Stripping TCK
JCP
TCK License
OPTIONAL PART OF TCK LICENSE
Has direct obligation to Spec Lead if Stripping is desired
Spec Implementer e.g., "Platform provider"
TCK
verifies
Implementer Binary e.g, "Platform"
EULA
Stripping TCK
Stripping Tool
Application Vendor (software only or bundled on/with hardware)
Stripped implementer Binary
App
Application Deliverable ("Sealed")
A Spec Implementer who wishes to allow stripping requires App Vendor to acquire TCK license from Spec Lead and pass the optional part of the TCK.

- Spec Implementer optionally requires App Vendor to optionally accept TCK license and test from Spec Lead if they wish to strip an implementation

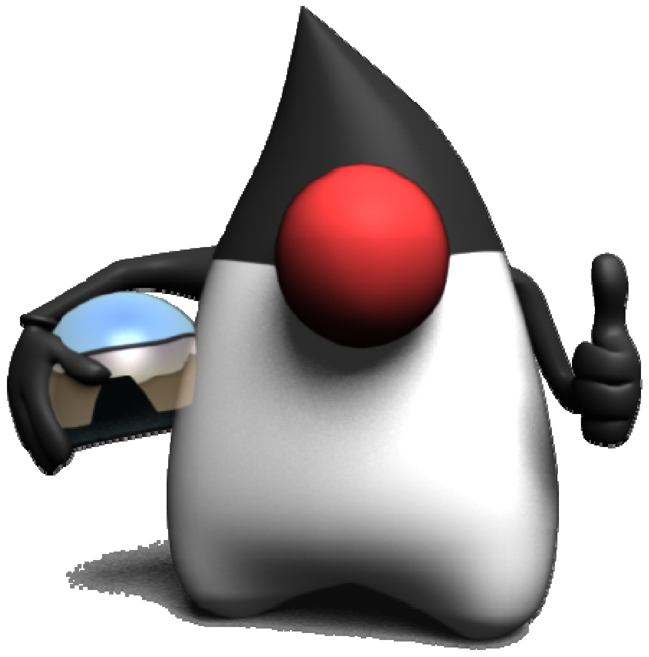
DRAFT – FOR DISCUSSION PURPOSES

Licensing Proposal – End User POV


Spec Lead
TCK
Stripping TCK
JCP
TCK License
OPTIONAL PART OF TCK LICENSE
Spec Implementer e.g., "Platform provider"
TCK
verifies
Implementer Binary e.g, "Platform"
EULA
Stripping TCK
Enterprise End User
Stripped implementer Binary
App
Stripping Tool
Has direct obligation to Spec Lead if Stripping is desired
Enterprise, typically stripping when running in their data center. Could also be done by a savvy home user / developer / hobbyist.
A Spec Implementer who wishes to allow stripping requires End User to acquire TCK license from Spec Lead and pass the optional part of the TCK.

- Spec Implementer optionally requires End User to optionally accept TCK license and test from Spec Lead if they wish to strip an implementation

Summary of Impact on Relevant Documents (1 of 2)



- JSPA – No changes required
- Specification License – No changes required
- Specification:
  - Define "Fully Implemented" and "Application Specific"
  - Add condition that, once stripped, implementations become "closed", aka "Sealed Application Bundle" (no further changes, no exposed APIs, etc)

Summary of Impact on Relevant Documents (2 of 2)

- TCK License
  - Creation of the TCK License related to stripping
  - Updates to allow downstream “stripping” upon condition of accepting Spec Lead’s “Optional part of TCK License”
- TCK
  - Addition of TCK related to Stripping
- Platform Provider (aka Spec Implementer’s) Binary License (e.g., the “BCL” for Oracle Implementations)
  - Updates to allow direct licensee “stripping” upon condition of accepting Spec Lead’s “Optional part of TCK License”

Java™