Java, OpenSAF and AM4J

Johan Mårtensson
System Architect, Ericsson

Johan.o.martensson@ericsson.com

Presentation Outline

- Java in SAF
- AM4J – Availability Management for Java
- AM4J and OpenSAF
- Demonstration

Presentation Outline

- Java in SAF
- AM4J – Availability Management for Java
- AM4J and OpenSAF
- Demonstration

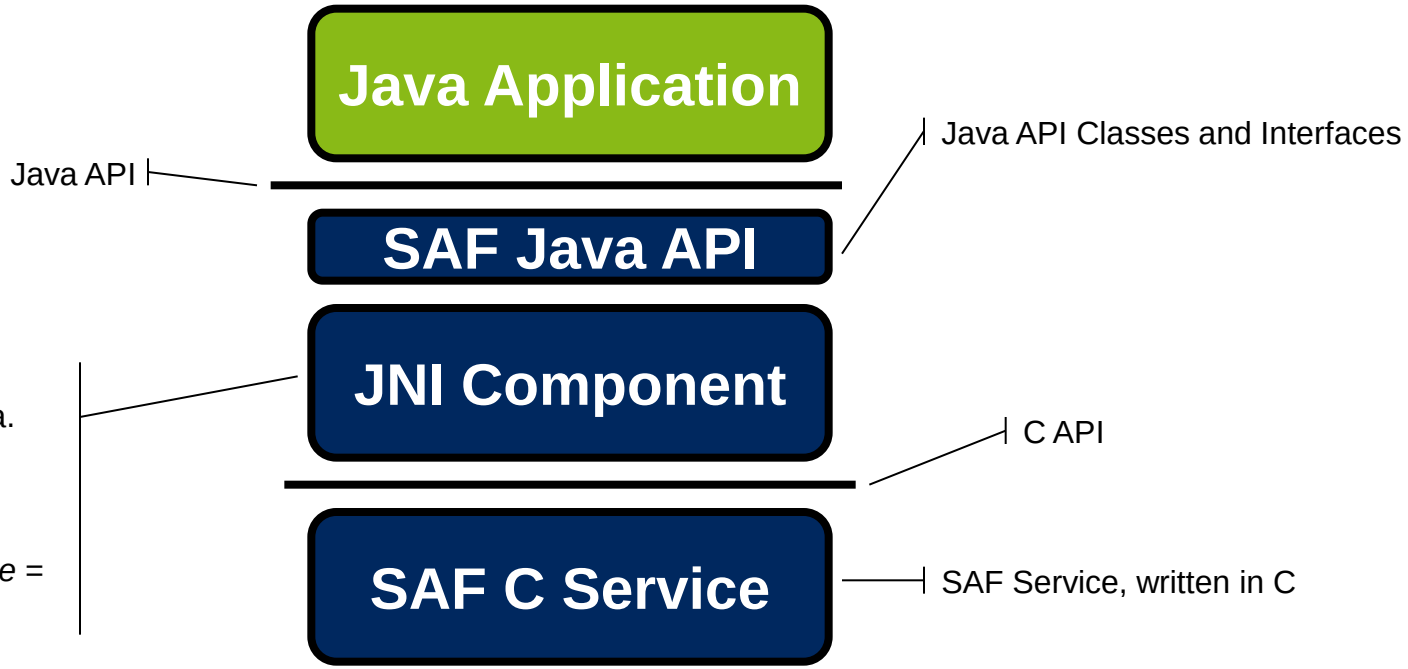
Why Java in SAF?

- Java
  - Java is common in server-side applications
  - Huge eco-system of Java components and frameworks
  - SIP servlets in Java makes it a good fit for telco
- Java and SAF
  - Nodes with C and Java mixed need unified O&M and Availability
  - No good standard O&M interface for application servers

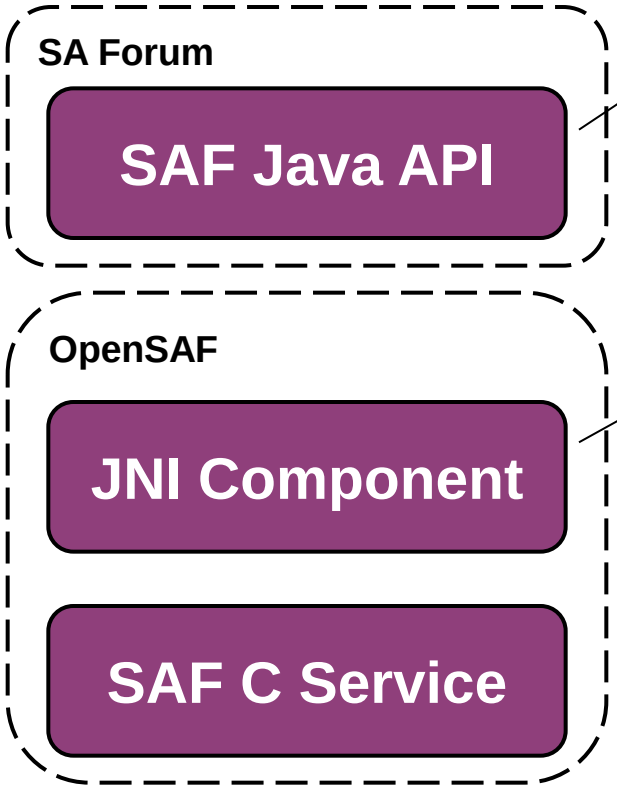
Java in SAF

- Java APIs are standardized in SA Forum
  - Maps the C API to a Java-style API
- Java APIs are defined for
  - AMF (B 04 01)
  - IMM (A 03 00 15)
  - NTF (A 03 01)
  - CLM (B 04 01)
  - EVT (B 03 01)

Java APIs and C Bindings



Java in SAF & OpenSAF



The API is defined in SA Forum. The API only contains the classes and interfaces that defines it towards an application.

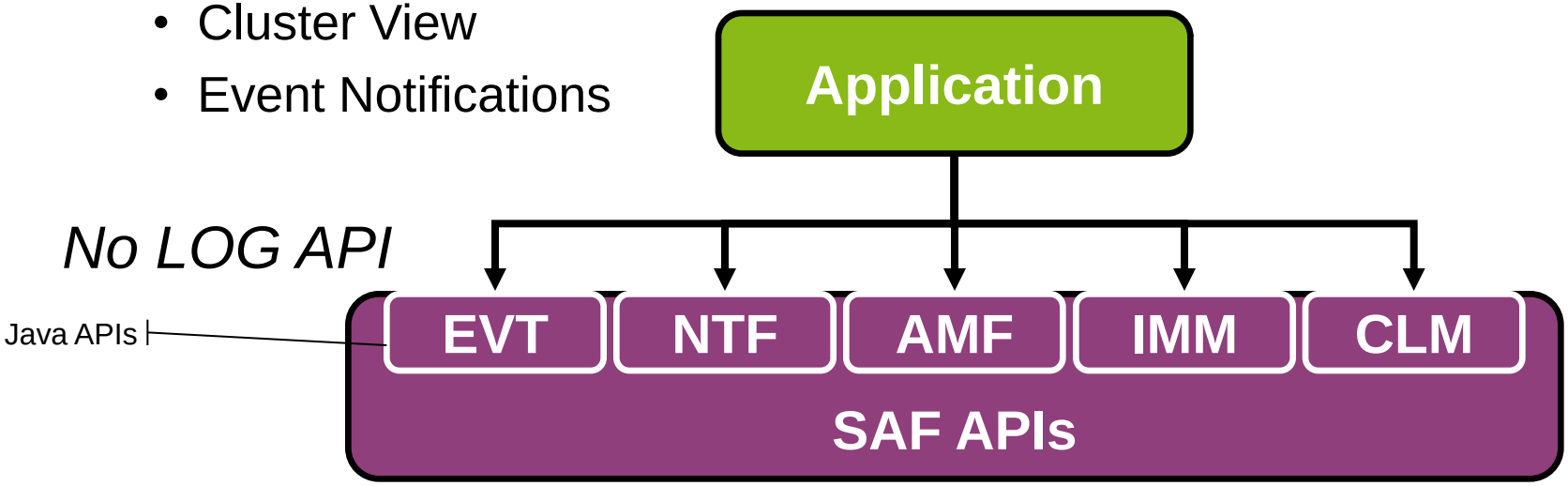
The JNI component implements the imported API. It is part of OpenSAF.

Java Application on SAF

Java Application on SAF gets

- High Availability
- Configuration Management
- Fault Management
- Cluster View
- Event Notifications

No LOG API



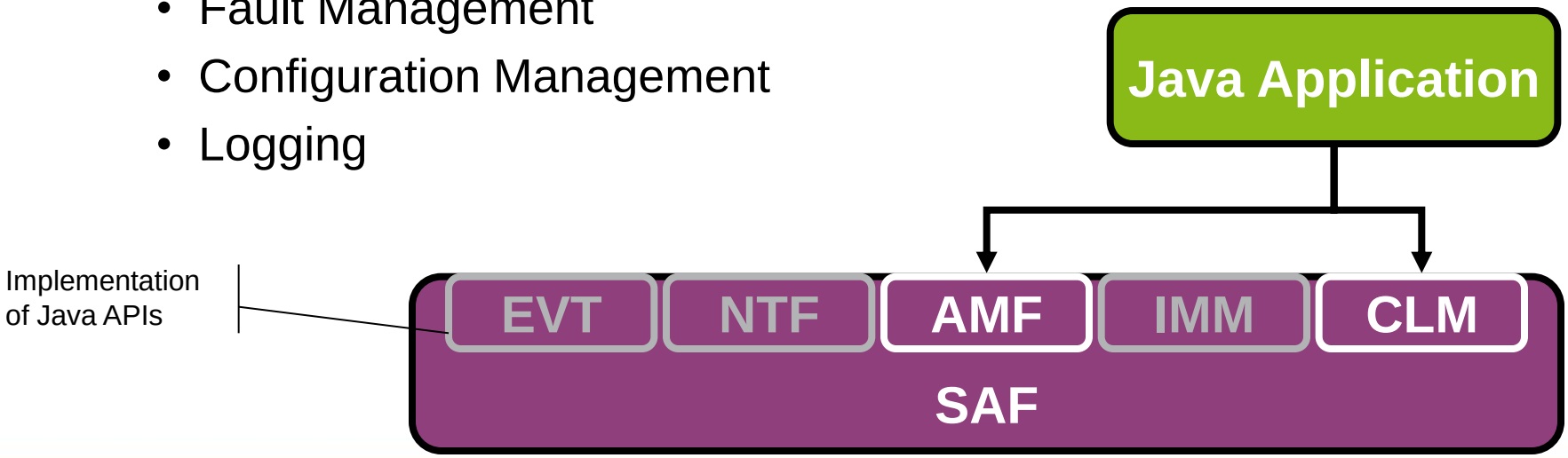
Java Application on OpenSAF

Java Application on OpenSAF gets

- High Availability
- Cluster View

Important missing pieces

- Fault Management
- Configuration Management
- Logging

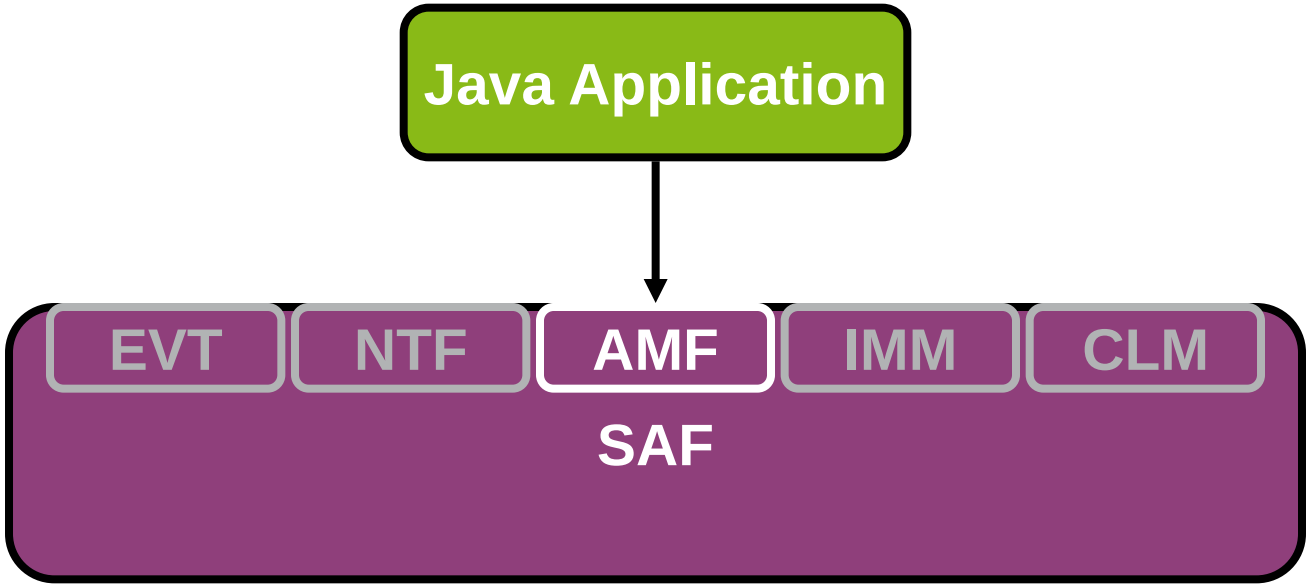


Need for High-Level APIs

- Java APIs in SAF are low-level and maps to C APIs
- Java on the server is mostly Java EE
- Typical Java EE APIs are high-level

In most cases, the Java APIs in SAF should not be used directly from Java EE applications

Availability



Presentation Outline

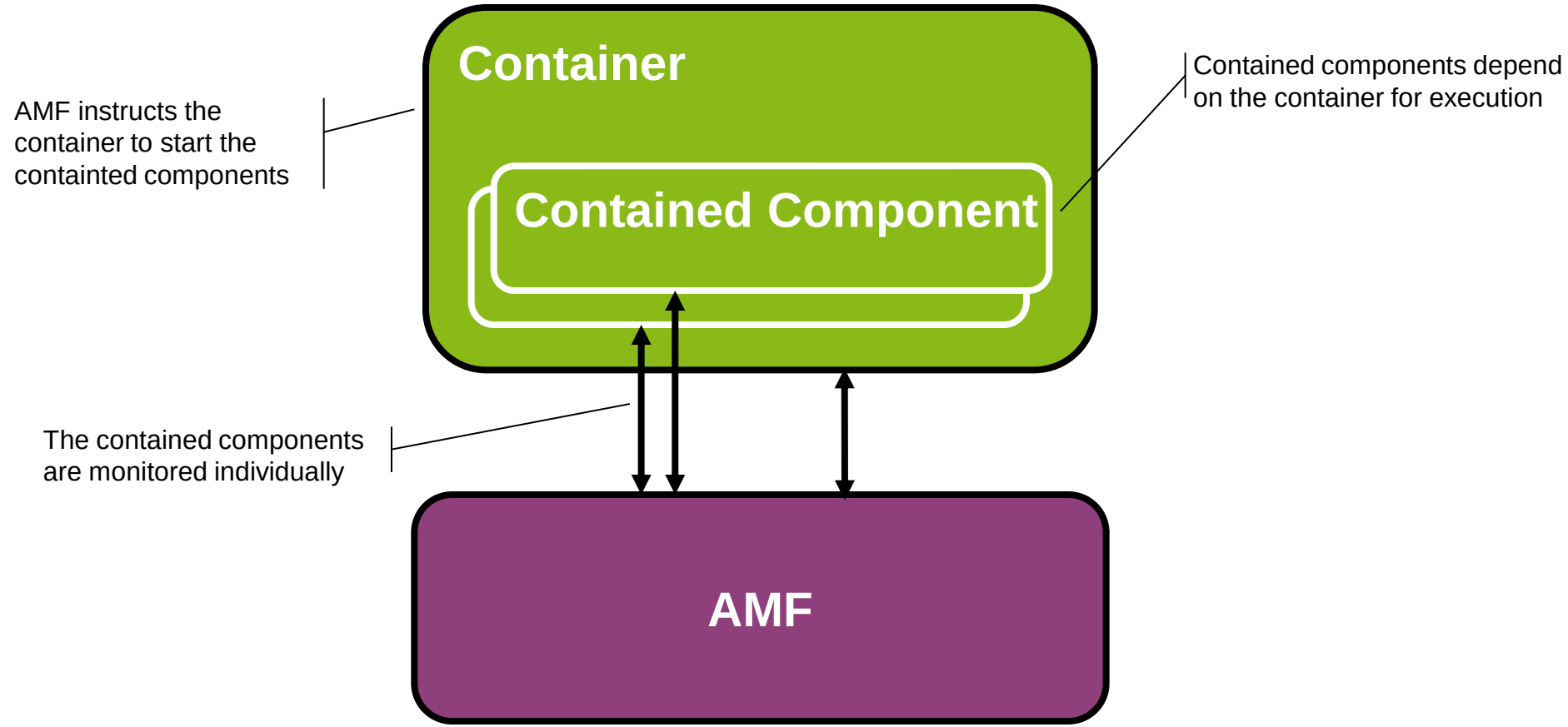
- Java in SAF
- AM4J – Availability Management for Java
- AM4J and OpenSAF
- Demonstration

Availability Management for Java (AM4J)

- Standardized availability management for Java
- Manages a container and its contained components
- Enables
  - Instantiate/cleanup
  - Activate/deactivate
  - Health checking
  - Error reports

  ... for the container and the contained components

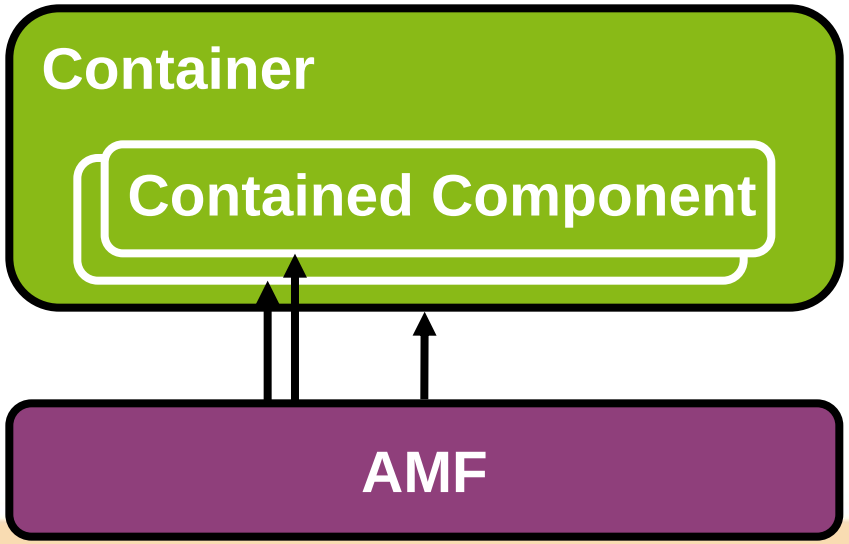
Container and Contained Components



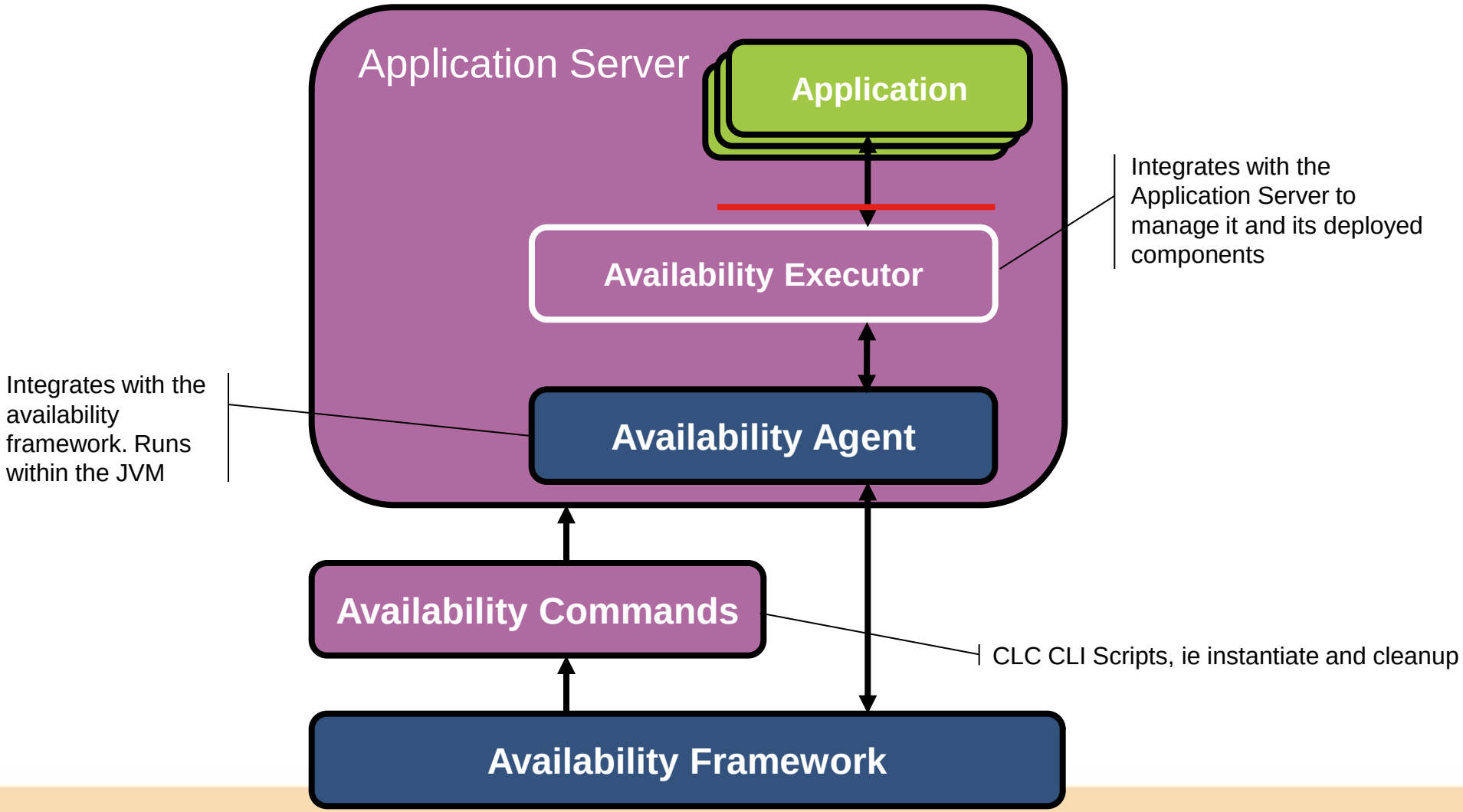
Container and Contained Components

- Contained components depend on the container for execution
- Container is responsible for instantiating/cleaning up contained components
- AMF supervises contained components directly

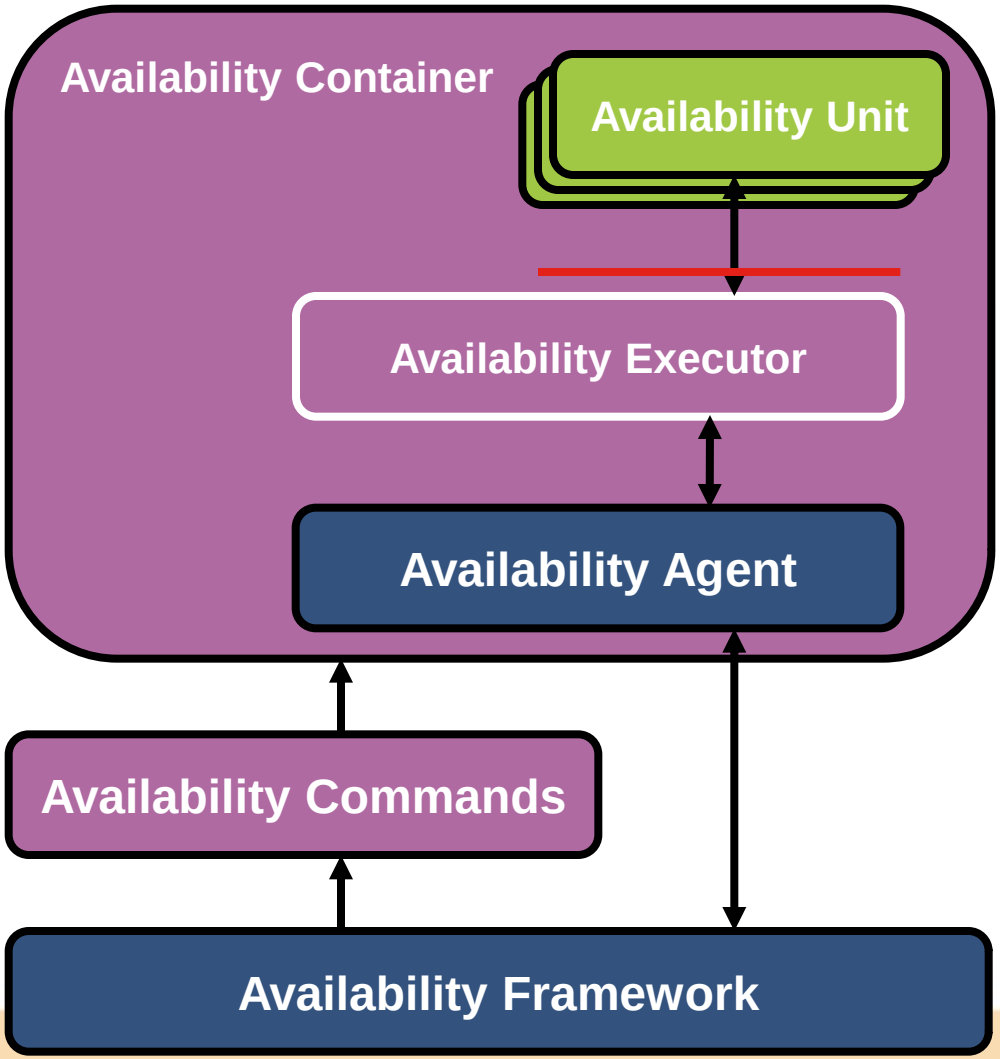
Not Java-specific



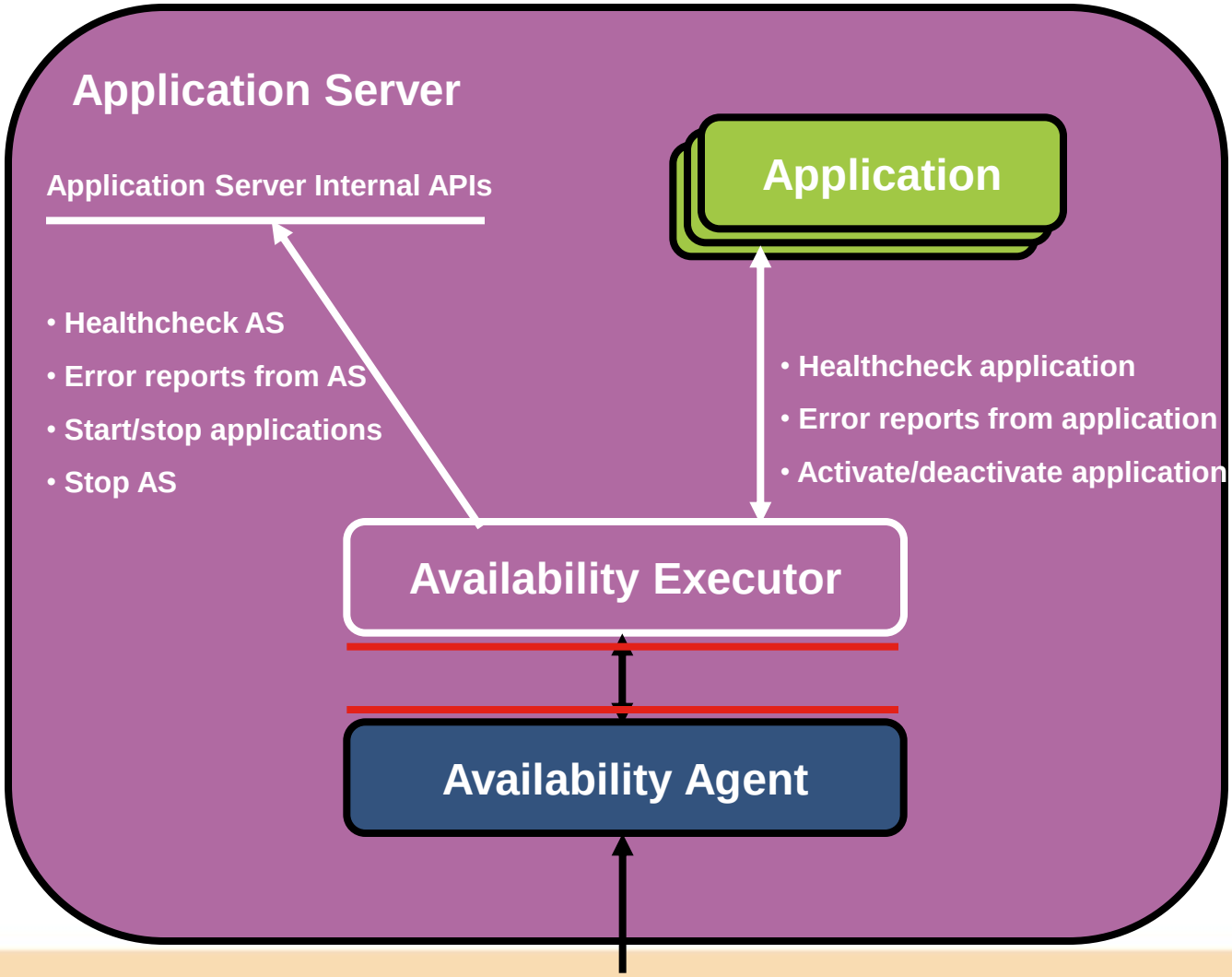
AM4J Architecture



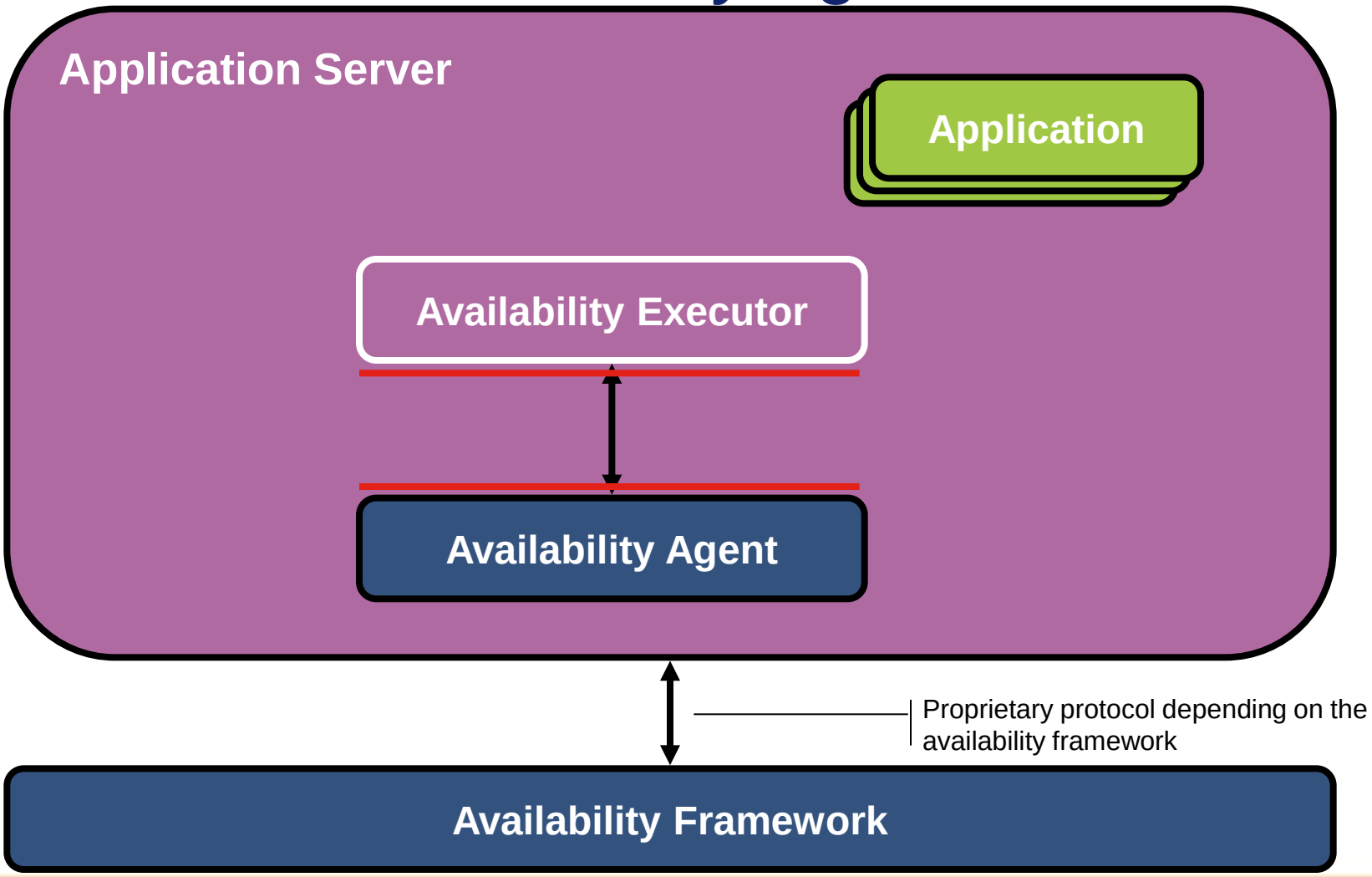
AM4J Terminology



Availability Executor



Availability Agent

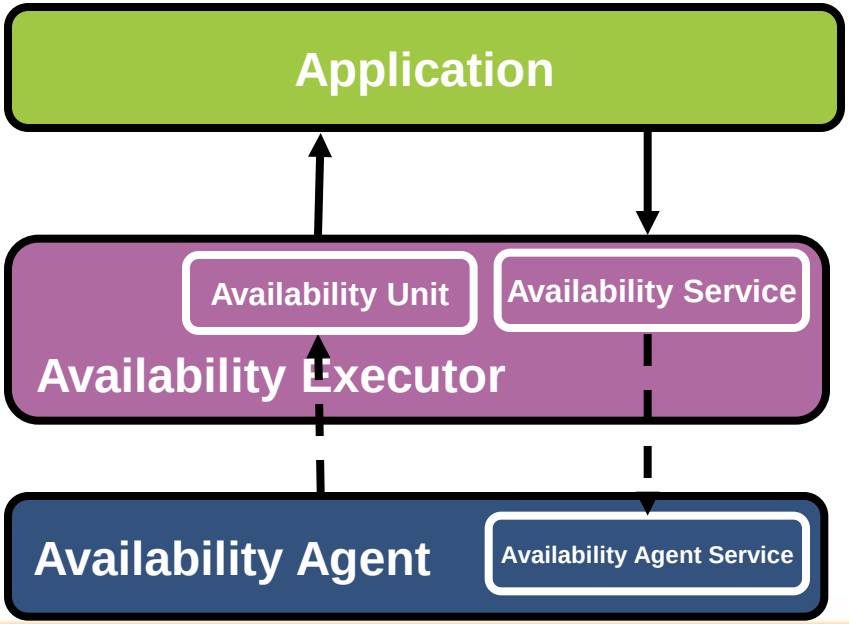


Application Integration

- No integration necessary in applications - 3rd part/legacy is OK
  - Gives instantiate/cleanup and activate/deactive integration
- Applications that use AM4J APIs get
  - Health checking
  - Error reporting
  - Activation/deactivation reason
  - Activation attributes – CSI Attributes

Application Interface

- **AvailabilityService** is the Application Interface. Set with dependency injection
- **AvailabilityService** communicates with the Availability Agent



Servlet Example

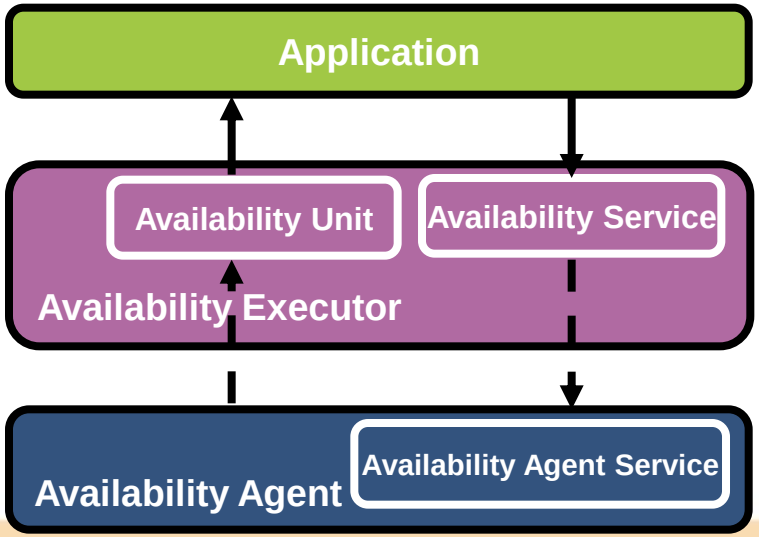
```
class MyServlet extends HttpServlet

  @Resource (name="availabilityService")
  private AvailabilityService service;

  public void init() {
    System.out.println("I am " + service.getName());
  }

  void doGet(...) {
    // ... Things go wrong
    service.reportError(ERROR_CANNOT_RECOVER);
  }

  @HealthCheck
  private HealthState checkHealth() {
    // ... Verify health
    return HEALTHY;
  }
}
```



AM4J Material

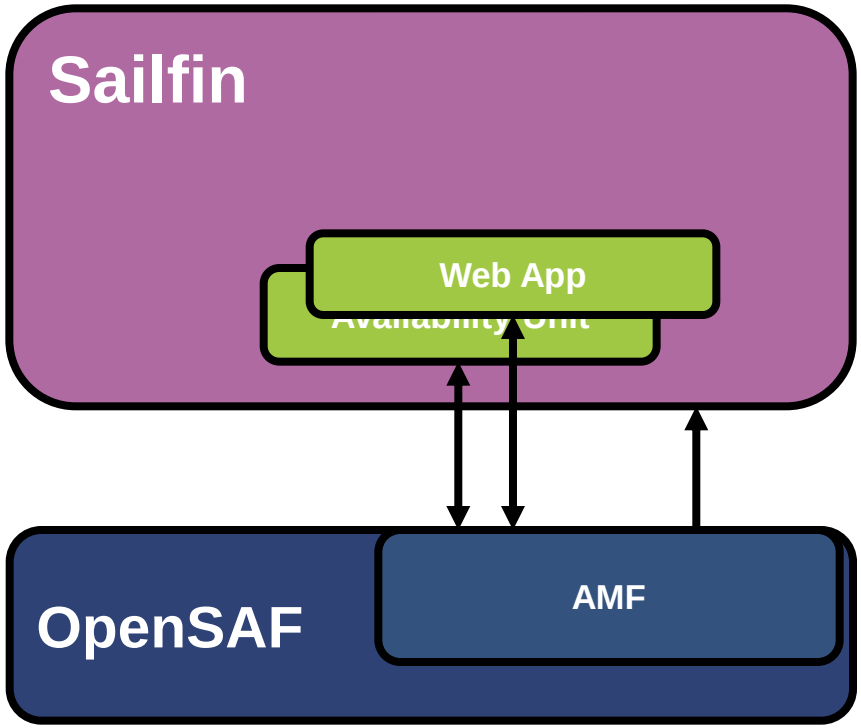
- Rerefence implementation with Sailfin
- Specification http://jcp.org/en/jsr/summary?id=319
- Open Source project
  - http://availabilitymanagement.dev.java.net
  - GPL v2 with Classpath exception

Presentation Outline

- Java in SAF
- AM4J – Availability Management for Java
- AM4J and OpenSAF
- Demonstration

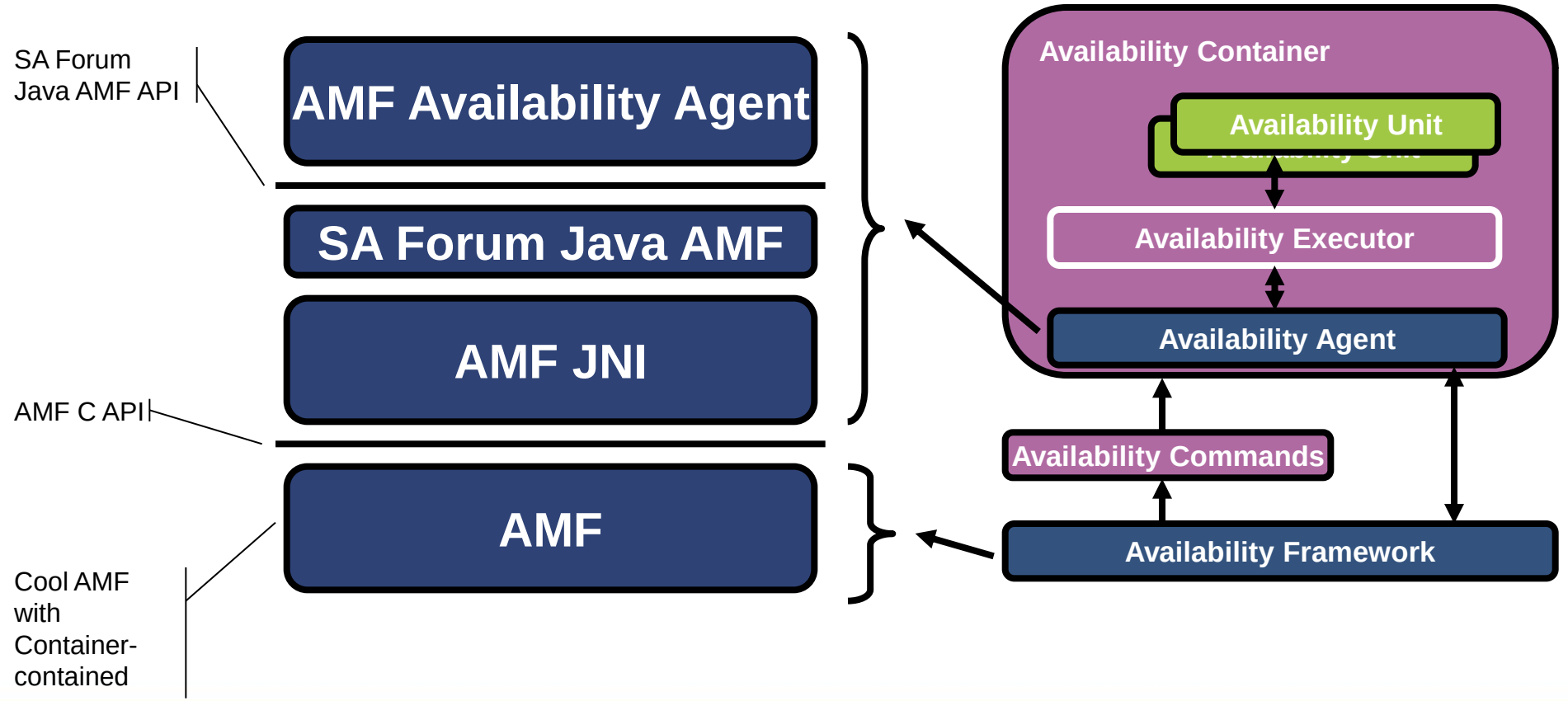
SAF and AM4J – Ideally

- AMF Manages both Sailfin and contained applications



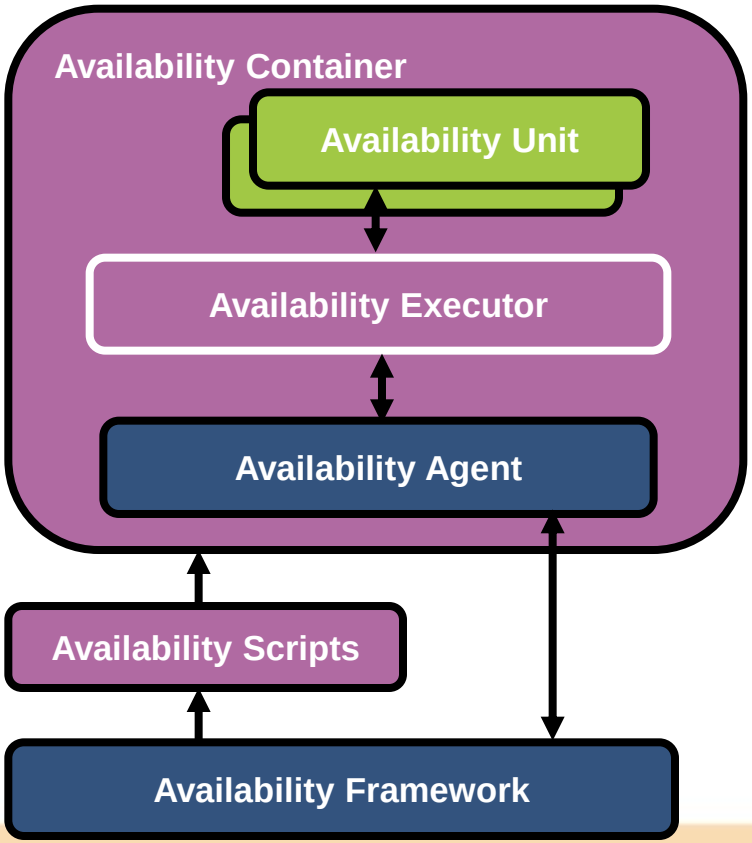
OpenSAF Implementation

OpenSAF AM4J & AMF Stack



OpenSAF TODO

- ✓ Has an AMF JNI implementation
- Lacks container-contained
- Lacks an Availability Agent for AMF



Java Wishlist

- OpenSAF
  - Java API for IMM
  - Java API for NTF
  - Java API for LOG
  - Container-contained support in AMF
- SAF
  - Standardized Java LOG API

Presentation Outline

- Java in SAF
- AM4J – Availability Management for Java
- AM4J and OpenSAF
- Demonstration

Demonstration

- Startup of Sailfin and Web app
- Fail-Over of Sailfin

- More exciting with Container-contained. Next year??

Start Up

1. Availability Container
   1. Availability Command INSTANTIATE is invoked
   2. Availability Container is initialized
   3. Availability Container is activated
2. Availability Unit
   1. The Availability Container is requested to instantiate an Availability Unit
   2. The Availability Unit is activated

Availability Container Fail-Over

The Availability Container is running on node A

1. The Availability Framework health checks the Availability Container
2. The Availability Container responds with an error
3. The Availability Framework terminates the Availability Container on node A
4. The Availability Framework instantiates the Availability Container on node B
5. The Availability Framework activates the Availability Container on node B

Questions ?

Thank You!

For more information:

http://devel.opensaf.org
http://availabilitymanagement.dev.java.net