**Share**

****

**You said:**

Given a context, Web Application using Turbine framework with Spring Framework 4.x/5.x and Java version 8 and running on Jboss 7.3(EAP 7.x) to JBoss 8.x (EAP 8.x) and IBM MQ is a key component to communicate with Down Stream system . Please guide line the Steps to achieve the MIgration with Minimal Effort

**ChatGPT said:**

Migrating a **Java-based web application** using the **Turbine framework**, **Spring 4.x/5.x**, and **IBM MQ**, running on **JBoss EAP 7.3 to JBoss EAP 8.x**, can be done with **minimal effort** if you follow a well-structured plan.

Here’s a **step-by-step guideline** to help you with the migration while minimizing risk and effort:

**🔧 1. Preparation Phase**

**✅ Inventory and Assessment**

* Document all:
  + External systems (like IBM MQ endpoints).
  + JNDI configurations, datasources, and queues.
  + Dependencies (Spring version, third-party libraries).
  + Application server configurations (standalone.xml, logging, etc.).
* Check compatibility of:
  + **Spring 4.x/5.x** with JBoss EAP 8.x.
  + **Java 8** (JBoss EAP 8 requires Java 11+ by default).

💡 **Note**: JBoss EAP 8.x supports Java 11+ officially, so Java 8 might **not** be viable long term.

**⚙️ 2. Environment Setup**

**🛠 Install JBoss EAP 8.x**

* Set up a parallel environment for JBoss 8.x (do not overwrite existing 7.3 setup).
* Install and configure:
  + **IBM MQ Resource Adapter**.
  + **JNDI entries** for connection factories and queues.

**🔍 Match Configuration**

* Match standalone.xml (or domain.xml) entries from JBoss 7.3 to JBoss 8.
  + JMS Connection Factories
  + Security Domains
  + Data Sources
  + Logging configuration
  + JVM memory settings

**🧱 3. Update Application Code/Build**

**🔗 Dependencies & Build Files**

* Check for deprecated APIs or libraries:
  + Some JBoss 7.x libraries (e.g., javax.*) might need to be replaced with jakarta.* (EAP 8 is Jakarta EE 10).
  + **Spring 4.x** may work, but Spring 5.x is safer.
* Replace any **JBoss-specific** classes that are deprecated.
* Update:
  + pom.xml or build.gradle to use correct JBoss and MQ dependencies.

**💼 Turbine Framework**

* Turbine is not commonly updated – test thoroughly in new container.
* Ensure any **Servlet, Filter, or Listener** definitions align with Jakarta EE if migrating APIs.

**🪄 4. IBM MQ Configuration (Critical Step)**

**IBM MQ Integration**

* Install the **IBM MQ Resource Adapter** in JBoss EAP 8 (deployments/ or modules/).
* Configure resource-adapters subsystem with the correct activationspec, connection-factory, and admin-objects.

**Sample in**standalone-full.xml**:**

xml

CopyEdit

<resource-adapters> <resource-adapter id="wmq.jmsra.rar"> <archive>wmq.jmsra.rar</archive> <transaction-support>XATransaction</transaction-support> <connection-definitions> <connection-definition class-name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl" jndi-name="java:/mq/ConnectionFactory" pool-name="MQConnectionFactoryPool"> <config-property name="hostName">mqhost</config-property> <config-property name="port">1414</config-property> <config-property name="channel">CHANNEL.NAME</config-property> <config-property name="queueManager">QMGR</config-property> <config-property name="transportType">CLIENT</config-property> </connection-definition> </connection-definitions> <admin-objects> <admin-object class-name="com.ibm.mq.connector.outbound.MQQueueProxy" jndi-name="java:/mq/MyQueue" pool-name="MQQueue"> <config-property name="baseQueueName">QUEUE.NAME</config-property> </admin-object> </admin-objects> </resource-adapter> </resource-adapters>

**🚀 5. Testing in JBoss EAP 8.x**

**Smoke Test the App**

* Deploy WAR to JBoss EAP 8.
* Validate:
  + Spring context loads correctly.
  + All MQ messages are sent/received properly.
  + All configuration resources are bound in JNDI as expected.

**Logging and Monitoring**

* Check logs for:
  + MQ connectivity issues
  + ClassNotFound or NoClassDefFound errors (usually due to module differences)
  + Spring-related bean loading failures

**🔄 6. Backward Compatibility**

* You can **run both JBoss 7.3 and 8.x in parallel** temporarily for:
  + Performance/load testing
  + Incremental cutover
* Implement toggle for message sending to either MQ instance if required

**📦 7. Deployment and Rollout**

* Plan deployment during low-traffic periods.
* Roll back strategy:
  + Keep JBoss 7.3 instance ready for fallback
  + Backup all MQ configurations

**📝 Summary Checklist**

| **Task** | **Status** |
| --- | --- |
| Assess existing app & dependencies | ✅ |
| Set up JBoss EAP 8 environment | ✅ |
| Port standalone.xml config | ✅ |
| Install IBM MQ RAR | ✅ |
| Update Spring & Turbine compatibility | ✅ |
| Modify build dependencies | ✅ |
| Configure JNDI & MQ resources | ✅ |
| Deploy WAR & verify MQ | ✅ |
| Perform regression tests | ✅ |
| Plan and execute go-live | ✅ |

Would you like a **sample JBoss 8 configuration file** or **example Spring MQ sender/receiver beans** tailored for this setup?

**You said:**

Please share different configuration files of EAP 7.3 amd configuration files of EAP 8.0 and show the differences on the Structure compare agaist each other

**ChatGPT said:**

Sure! Here's a **side-by-side structural comparison** of the configuration files between **JBoss EAP 7.3** and **JBoss EAP 8.0**, focusing on the **key subsystems** and changes, especially relevant for your context (Spring app with IBM MQ and Turbine on Java 8+).

**🗂 Configuration Files Overview**

| **EAP Version** | **Main Config Files** | **Default Profile** |
| --- | --- | --- |
| EAP 7.3 | standalone.xml, standalone-full.xml, domain.xml | standalone-full.xml (for JMS) |
| EAP 8.0 | standalone.xml, standalone-full.xml, domain.xml | standalone-full.xml (Jakarta EE 10) |

📝 We'll compare standalone-full.xml, as it includes full JMS, EE, and web capabilities required for IBM MQ and Spring-based applications.

**🔍 Key Differences in Structure: EAP 7.3 vs EAP 8.0**

**1. 📦 Namespaces and Schema Versions**

**✅ EAP 7.3**

xml

CopyEdit

<server xmlns="urn:jboss:domain:4.0"> <extensions> <extension module="org.jboss.as.messaging"/> <extension module="org.jboss.as.ee"/> ... </extensions>

**✅ EAP 8.0**

xml

CopyEdit

<server xmlns="urn:jboss:domain:26.0"> <extensions> <extension module="org.wildfly.messaging-activemq"/> <extension module="org.wildfly.ee"/> ... </extensions>

✅ **Diff**: org.jboss.as.\* modules renamed to org.wildfly.\* in EAP 8.0 (reflects WildFly base). Schema versions (from 4.0 to 26.0) have changed significantly.

**2. 🧩 EE Subsystem**

**📄 EAP 7.3**

xml

CopyEdit

<ee xmlns="urn:jboss:domain:ee:4.0"> <global-modules> <module name="com.ibm.mq.allclient" slot="main"/> </global-modules> </ee>

**📄 EAP 8.0**

xml

CopyEdit

<ee xmlns="urn:jboss:domain:ee:9.0"> <annotation-property-replacement>false</annotation-property-replacement> <global-modules> <module name="com.ibm.mq.allclient" slot="main" services="export"/> </global-modules> </ee>

✅ **Diff**: EAP 8.0 includes stricter EE config and more refined modular controls (services="export" for module usage across apps).

**3. ✉️ Messaging Subsystem**

If you're **not using ActiveMQ** but rather **IBM MQ as a Resource Adapter**, this still matters for base support.

**📄 EAP 7.3 (HornetQ/ActiveMQ Artemis)**

xml

CopyEdit

<messaging-activemq xmlns="urn:jboss:domain:messaging-activemq:4.0"> ... </messaging-activemq>

**📄 EAP 8.0 (Updated Artemis)**

xml

CopyEdit

<messaging-activemq xmlns="urn:jboss:domain:messaging-activemq:18.0"> ... </messaging-activemq>

✅ **Diff**: The structure remains similar, but schema and defaults are stricter in EAP 8. You’ll likely ignore this if using **IBM MQ Resource Adapter** instead.

**4. 🔄 Resource Adapters (IBM MQ Configuration)**

**📄 EAP 7.3**

xml

CopyEdit

<resource-adapters xmlns="urn:jboss:domain:resource-adapters:5.0"> <resource-adapter> <archive>wmq.jmsra.rar</archive> <connection-definitions> <connection-definition class-name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl" jndi-name="java:/mq/ConnectionFactory"> ... </connection-definition> </connection-definitions> </resource-adapter> </resource-adapters>

**📄 EAP 8.0**

xml

CopyEdit

<resource-adapters xmlns="urn:jboss:domain:resource-adapters:9.0"> <resource-adapter id="wmq.jmsra.rar"> <archive>wmq.jmsra.rar</archive> <transaction-support>XATransaction</transaction-support> <connection-definitions> <connection-definition class-name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl" jndi-name="java:/mq/ConnectionFactory"> ... </connection-definition> </connection-definitions> </resource-adapter> </resource-adapters>

✅ **Diff**:

* Schema changed from 5.0 → 9.0.
* More explicit IDs and naming conventions recommended in EAP 8.
* Configuration style remains **mostly compatible**.

**5. 🛡️ Security Subsystem**

**📄 EAP 7.3**

xml

CopyEdit

<security-domains> <security-domain name="other" cache-type="default"> <authentication> <login-module code="Remoting" flag="optional"/> </authentication> </security-domain> </security-domains>

**📄 EAP 8.0**

xml

CopyEdit

<elytron> <security-domains> <security-domain name="ApplicationDomain" default-realm="ApplicationRealm"> ... </security-domain> </security-domains> </elytron>

✅ **Diff**:

* **Elytron** is **mandatory** in EAP 8; PicketBox is deprecated.
* You'll need to migrate security domains, users, and roles to Elytron config.

**6. 🔧 Module Paths & Classloading**

EAP 8 is stricter about **module isolation**:

* You must explicitly expose modules using services="export" or use <dependencies> in jboss-deployment-structure.xml.

**✅ Summary: What Needs Attention During Migration**

| **Area** | **Change Required?** | **Notes** |
| --- | --- | --- |
| Schema versions | ✅ Yes | Must update namespaces in XML |
| IBM MQ RAR config | 🔄 Minor | Mostly compatible with updated namespace |
| EE Subsystem | 🔄 Minor | Add services="export" for global modules |
| Messaging Subsystem | ❌ No (for IBM MQ) | Used only if you use embedded Artemis |
| Security (PicketBox → Elytron) | ✅ Major | Migrate to Elytron |
| Module/Classloading | ✅ Yes | Add jboss-deployment-structure.xml |
| Logging | ✅ Optional | Review logging subsystem schema differences |
| Java 8 Compatibility | ⚠️ Deprecated | JBoss EAP 8 **requires Java 11+** |

Would you like:

1. A **side-by-side diff file** (actual XML content)?
2. A **sample**standalone-full.xml**for EAP 8** pre-configured with IBM MQ?
3. Help with converting your **EAP 7**security**to Elytron**?

Let me know what you'd like to dig into next.

**You said:**

A side by side diff file

**ChatGPT said:**

Eap7 Vs Eap8 Config Diff

I've created a side-by-side diff XML file comparing key sections of standalone-full.xml from **JBoss EAP 7.3** and **EAP 8.0**. It highlights changes in:

* Namespace and schema versions
* EE subsystem
* IBM MQ resource adapter
* Security (PicketBox vs Elytron)
* Messaging subsystem

Let me know if you want:

* A complete working standalone-full.xml for EAP 8 pre-configured for IBM MQ
* Help with Elytron configuration migration
* A checklist or script for automated migration steps

**You said:**

A complete working standalone-full.xml for EAP 8 pre-configured for IBM MQ Help with Elytron configuration migration A checklist or script for automated migration steps

**ChatGPT said:**

Eap7 Vs Eap8 Config Diff

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

<!-- Side-by-side comparison: JBoss EAP 7.3 vs EAP 8.0 standalone-full.xml excerpts -->

<!-- ========================= -->

<!-- Section: Root & Namespace -->

<!-- ========================= -->

<!-- EAP 7.3 -->

<server xmlns="urn:jboss:domain:4.0">

<extensions>

<extension module="org.jboss.as.ee"/>

<extension module="org.jboss.as.messaging"/>

<extension module="org.jboss.as.connector"/>

<!-- other extensions -->

</extensions>

<!-- EAP 8.0 -->

<server xmlns="urn:jboss:domain:26.0">

<extensions>

<extension module="org.wildfly.ee"/>

<extension module="org.wildfly.messaging-activemq"/>

<extension module="org.wildfly.connector"/>

<!-- other extensions -->

</extensions>

<!-- ======================== -->

<!-- Section: EE Subsystem -->

<!-- ======================== -->

<!-- EAP 7.3 -->

<ee xmlns="urn:jboss:domain:ee:4.0">

<global-modules>

<module name="com.ibm.mq.allclient" slot="main"/>

</global-modules>

</ee>

Edited

Here’s what you now have:

**✅ Complete**standalone-full.xml**for EAP 8 with IBM MQ configured**

**🔐 Elytron Migration Help**

EAP 8 uses Elytron, replacing PicketBox. Here's a guide for migrating:

**1. Identify existing security domains**

From EAP 7.3:

xml

CopyEdit

<security-domain name="other"> <authentication> <login-module code="Remoting" flag="optional"/> </authentication> </security-domain>

**2. Translate to Elytron in EAP 8:**

xml

CopyEdit

<subsystem xmlns="urn:jboss:domain:elytron:10.0"> <security-domains> <security-domain name="ApplicationDomain" default-realm="ApplicationRealm"> <realms> <realm name="ApplicationRealm"/> </realms> </security-domain> </security-domains> <realms> <identity-realm name="ApplicationRealm"> <identity> <attribute name="role">admin</attribute> </identity> </identity-realm> </realms> </subsystem>

**✅ Migration Checklist (Minimal Downtime)**

**✅ Pre-Migration**

* Back up full JBoss EAP 7.3 installation and domain configuration.
* Export IBM MQ JNDI/resource adapter configs.
* Review app dependencies (Spring, JMS, MQ client versions).
* Ensure app is compatible with Java 11 (for EAP 8).

**✅ Migration Steps**

1. **Install JBoss EAP 8.0**
   * Download and extract to a new location.
   * Set JAVA\_HOME to Java 11.
2. **Migrate Configs**
   * Replace standalone-full.xml with the modified version.
   * Add IBM MQ RAR file in <EAP\_HOME>/standalone/deployments.
3. **Update Modules**
   * Copy IBM MQ module JARs into modules/com/ibm/mq/allclient/main.
   * Add a module.xml for MQ:

xml

CopyEdit

<module xmlns="urn:jboss:module:1.5" name="com.ibm.mq.allclient"> <resources> <resource-root path="com.ibm.mq.allclient.jar"/> </resources> </module>

1. **Configure Elytron**
   * Replace PicketBox with Elytron as shown above.
2. **Start Server and Validate**
   * bin/standalone.sh -c standalone-full.xml
   * Check MQ connectivity.
   * Test deployed app.

**✅ Post-Migration**

* Load testing
* Failover testing
* Backup of EAP 8 configs

Would you like:

* A shell script to automate deployment and backup steps?
* A module.xml for MQ module pre-generated?
* JNDI test utility to validate MQ connection from EAP 8?

Let me know!

**You said:**

shell script to automate deployment and backup steps? A module.xml for MQ module pre-generated? JNDI test utility to validate MQ connection from EAP 8?

**ChatGPT said:**

Perfect! Below are the three items you requested for simplifying and automating the migration.

**✅ 1. Shell Script to Automate Backup, Deployment, and IBM MQ Setup**

bash

CopyEdit

#!/bin/bash # Set variables EAP\_OLD\_HOME="/opt/jboss-eap-7.3" EAP\_NEW\_HOME="/opt/jboss-eap-8.0" BACKUP\_DIR="/opt/backup/$(date +%Y%m%d\_%H%M%S)" MQ\_MODULE\_DIR="$EAP\_NEW\_HOME/modules/com/ibm/mq/allclient/main" RAR\_FILE="wmq.jmsra.rar" DEPLOY\_DIR="$EAP\_NEW\_HOME/standalone/deployments" # Step 1: Backup EAP 7.3 config echo "Backing up JBoss EAP 7.3 configuration to $BACKUP\_DIR..." mkdir -p "$BACKUP\_DIR" cp -r "$EAP\_OLD\_HOME/standalone/configuration" "$BACKUP\_DIR" # Step 2: Copy IBM MQ rar and create modules echo "Setting up IBM MQ module..." mkdir -p "$MQ\_MODULE\_DIR" cp ./com.ibm.mq.allclient.jar "$MQ\_MODULE\_DIR/" cp ./module.xml "$MQ\_MODULE\_DIR/" # Step 3: Deploy IBM MQ RAR echo "Deploying $RAR\_FILE to $DEPLOY\_DIR..." cp "./$RAR\_FILE" "$DEPLOY\_DIR/" touch "$DEPLOY\_DIR/$RAR\_FILE.dodeploy" # Step 4: Deploy customized configuration echo "Deploying new configuration file..." cp ./standalone-full.xml "$EAP\_NEW\_HOME/standalone/configuration/" echo "Migration Preparation Completed!"

✅ Customize paths as needed (especially JAR/RAR locations).  
✅ Place this script and all files in a staging folder before execution.

**✅ 2. Pre-generated**module.xml**for IBM MQ**

xml

CopyEdit

<module xmlns="urn:jboss:module:1.5" name="com.ibm.mq.allclient"> <resources> <resource-root path="com.ibm.mq.allclient.jar"/> </resources> <dependencies> <module name="javax.api"/> <module name="javax.jms.api"/> </dependencies> </module>

💡 This goes in:  
/opt/jboss-eap-8.0/modules/com/ibm/mq/allclient/main/module.xml

**✅ 3. JNDI Test Utility for MQ Connection (Java Program)**

java

CopyEdit

import javax.jms.\*; import javax.naming.Context; import javax.naming.InitialContext; import java.util.Properties; public class MQJndiTest { public static void main(String[] args) throws Exception { Properties props = new Properties(); props.put(Context.INITIAL\_CONTEXT\_FACTORY, "org.wildfly.naming.client.WildFlyInitialContextFactory"); props.put(Context.PROVIDER\_URL, "http-remoting://localhost:8080"); Context ctx = new InitialContext(props); ConnectionFactory cf = (ConnectionFactory) ctx.lookup("java:/mq/ConnectionFactory"); try (Connection connection = cf.createConnection()) { Session session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE); System.out.println("✅ MQ Connection established successfully!"); } catch (Exception e) { e.printStackTrace(); System.out.println("❌ MQ Connection failed."); } } }

**✅ Compile & Run:**

bash

CopyEdit

javac MQJndiTest.java java -cp .:jboss-client.jar MQJndiTest

Would you like a ZIP of all these pre-packaged? Or a Maven project for the JNDI utility?

**You said:**

given a scenario, where web applicationd eployed in Jboss 7.3 which all locations to be referred to see all the jar files being effective in making teh deplyed application working to revieww all those jars are compatible with EAP 8.0/Jboss8.0 to do preparatory work on libraries required to be upgrades

**ChatGPT said:**

Great question! When preparing to migrate a web application from **JBoss EAP 7.3 to EAP 8.0**, it’s critical to **review all JARs that the app depends on**, whether they come from:

* The application’s WAR/EAR
* The JBoss modules
* Deployment overlays
* Server classpath or bundled libraries

Here’s a **comprehensive checklist** of all locations where JARs may reside and be loaded from, so you can audit them for **Java 11** and **EAP 8 compatibility**.

**✅ 1. Application JARs (Packaged in WAR/EAR)**

📂 Typical location:

* $EAP\_HOME/standalone/deployments/yourapp.war/WEB-INF/lib/
* $EAP\_HOME/standalone/deployments/yourapp.ear/lib/

🛠️ **Action:**

* Extract the archive and list all JARs:

bash

CopyEdit

unzip -l yourapp.war | grep '\.jar'

* Use jdeps or jdeprscan for Java 11 compatibility check.

**✅ 2. JBoss Modules Used by the Application**

📂 Location:

* $EAP\_HOME/modules/

🔍 Includes:

* Shared modules (e.g., javax.api, org.apache.commons, custom modules)
* Resource Adapters (like IBM MQ)

🛠️ **Action:**

* Look in your standalone.xml for <global-modules> or any <module> references:

xml

CopyEdit

<global-modules> <module name="com.company.logging" slot="main"/> </global-modules>

* Check module path:

bash

CopyEdit

find $EAP\_HOME/modules -name "\*.jar"

**✅ 3. JBoss System/Runtime Libraries**

📂 Internal JARs:

* $EAP\_HOME/modules/system/layers/base/
* EAP internal modules like:
  + org.jboss.as.controller
  + org.jboss.logging
  + javax.servlet.api

🛠️ **Action:**

* These should not be used directly by applications.
* If your app relies on internal APIs (e.g., org.jboss.\*), it must be refactored.

**✅ 4. Deployment Overlays (if used)**

📂 Location:

* $EAP\_HOME/standalone/deployments/.overlays/

🛠️ **Action:**

* These can override JARs in deployed apps. Run:

bash

CopyEdit

find $EAP\_HOME/standalone/data/content -name "\*.jar"

**✅ 5. Custom Class Loaders or Added via JVM Args**

🔍 App or scripts may dynamically add classpath entries:

* -Djava.ext.dirs=...
* -cp in shell scripts

🛠️ **Action:**

* Check standalone.conf or startup scripts.
* Inspect app logs for ClassLoader behavior if custom loaders used.

**✅ 6. JNDI-Based or Resource Adapter JARs**

📂 Example: IBM MQ

* $EAP\_HOME/standalone/deployments/wmq.jmsra.rar
* Modules like: /modules/com/ibm/mq/allclient/main/

🛠️ **Action:**

* Check RAR contents: unzip -l wmq.jmsra.rar
* Check all JARs in the MQ module directory.

**✅ 7. Maven BOM Comparison for EAP 7.3 vs 8.0**

Use EAP BOMs to compare provided library versions:

xml

CopyEdit

<!-- EAP 7.3 BOM --> <dependency> <groupId>org.jboss.bom</groupId> <artifactId>jboss-eap-javaee7-with-tools</artifactId> <version>7.3.0.GA</version> <type>pom</type> </dependency> <!-- EAP 8 BOM --> <dependency> <groupId>org.jboss.bom</groupId> <artifactId>jboss-eap-ee-with-tools</artifactId> <version>8.0.0.GA</version> <type>pom</type> </dependency>

**🔎 Recommended Tooling**

* [jdeps](https://docs.oracle.com/en/java/javase/11/tools/jdeps.html): checks Java module dependencies
* [jdeprscan](https://docs.oracle.com/en/java/javase/11/tools/jdeprscan.html): detects deprecated APIs
* [mvn dependency:tree](https://maven.apache.org/plugins/maven-dependency-plugin/tree-mojo.html): for Maven apps

**⚠️ Common Compatibility Risks**

| **Area** | **Issue** |
| --- | --- |
| Java 11 Migration | Removed javax.xml.bind, javax.activation, etc. |
| Logging APIs | Switch to jboss-logging or SLF4J |
| Deprecated JEE modules | JSP, JSTL, EJB Lite usage |
| Old Spring versions | Update to Spring 5.x+ for Java 11 support |
| Commons/Apache JARs | Must match EAP 8-provided or relocate |

Would you like a script that:

* **Lists and extracts all JARs from deployed apps + modules**
* **Runs**jdeps**and flags Java 11 incompatibilities?**

**You said:**

Lists and extracts all JARs from deployed apps + modules Runs jdeps and flags Java 11 incompatibilities?

**ChatGPT said:**

Here’s a **complete shell script** that will:

1. 🔍 **List and extract all JARs** from:
   * Deployed apps (WAR/EAR)
   * Custom JBoss modules
2. ✅ **Run**jdeps on all JARs
3. 🚩 **Flag Java 11 incompatibilities**

**✅ Shell Script:**analyze-jars-java11.sh

bash

CopyEdit

#!/bin/bash # Configuration EAP\_HOME="/opt/jboss-eap-7.3" OUTPUT\_DIR="./jar\_analysis" JAVA\_VERSION="11" mkdir -p "$OUTPUT\_DIR/jars" mkdir -p "$OUTPUT\_DIR/reports" echo "🔍 Scanning deployments..." find "$EAP\_HOME/standalone/deployments" -name "\*.war" -or -name "\*.ear" | while read archive; do echo "📦 Extracting JARs from: $archive" unzip -q "$archive" "\*.jar" -d "$OUTPUT\_DIR/jars" done echo "🔍 Scanning modules..." find "$EAP\_HOME/modules" -name "\*.jar" -exec cp {} "$OUTPUT\_DIR/jars/" \; # Optionally scan MQ RAR file if [ -f "$EAP\_HOME/standalone/deployments/wmq.jmsra.rar" ]; then echo "🔍 Extracting MQ RAR JARs..." unzip -q "$EAP\_HOME/standalone/deployments/wmq.jmsra.rar" "\*.jar" -d "$OUTPUT\_DIR/jars" fi # Run jdeps for Java 11 compatibility check echo "✅ Running jdeps for Java $JAVA\_VERSION compatibility..." for jar in "$OUTPUT\_DIR/jars/"\*.jar; do base=$(basename "$jar") report="$OUTPUT\_DIR/reports/${base%.jar}\_jdeps.txt" echo "📋 Analyzing $base..." jdeps --multi-release "$JAVA\_VERSION" --jdk-internals "$jar" > "$report" 2>&1 done # Summary of potential issues echo "🚩 Scanning for internal API usage (may break in Java 11)..." grep -H "JDK internal API" "$OUTPUT\_DIR/reports/"\* | tee "$OUTPUT\_DIR/jdk-internal-usage-summary.txt" echo "✅ Analysis complete. See:" echo " - Extracted JARs: $OUTPUT\_DIR/jars" echo " - jdeps reports: $OUTPUT\_DIR/reports" echo " - Summary: $OUTPUT\_DIR/jdk-internal-usage-summary.txt"

**✅ Requirements:**

* JDK 11 installed (jdeps available in PATH)
* Permissions to access $EAP\_HOME

**✅ Output:**

* Extracted JARs: ./jar\_analysis/jars
* Jdeps report per JAR: ./jar\_analysis/reports/\*.txt
* Summary of Java 11 risky usage: ./jar\_analysis/jdk-internal-usage-summary.txt

Would you like this packaged as a **Docker container** or want it extended to:

* Check for deprecated Spring libraries?
* Auto-suggest compatible versions from Maven Central?

**You said:**

Check for deprecated Spring libraries? Auto-suggest compatible versions from Maven Central?

**ChatGPT said:**

Awesome! Let’s level this up. Below is an **enhanced shell + Python hybrid tool** that will:

**✅ Enhanced Features**

**1. 🔍 Scan JARs for:**

* **Deprecated Spring libraries**
* **Known legacy Apache Commons libs**
* **Old Jackson, Hibernate, JAXB, etc.**

**2. 📦 Look up compatible versions on Maven Central**

* Uses the Maven Central REST API

**🧩 Folder Structure**

pgsql

CopyEdit

migration-check/

├── analyze-jars-java11.sh <-- Shell driver script

├── check-legacy-libs.py <-- Python Maven compatibility checker

├── jar\_analysis/ <-- Output dir

**✅ 1. Shell Driver:**analyze-jars-java11.sh

bash

CopyEdit

#!/bin/bash EAP\_HOME="/opt/jboss-eap-7.3" OUTPUT\_DIR="./jar\_analysis" mkdir -p "$OUTPUT\_DIR/jars" echo "🔍 Extracting JARs from deployments and modules..." find "$EAP\_HOME/standalone/deployments" -name "\*.war" -or -name "\*.ear" | while read archive; do unzip -q "$archive" "\*.jar" -d "$OUTPUT\_DIR/jars" done find "$EAP\_HOME/modules" -name "\*.jar" -exec cp {} "$OUTPUT\_DIR/jars/" \; if [ -f "$EAP\_HOME/standalone/deployments/wmq.jmsra.rar" ]; then unzip -q "$EAP\_HOME/standalone/deployments/wmq.jmsra.rar" "\*.jar" -d "$OUTPUT\_DIR/jars" fi echo "✅ JARs ready in $OUTPUT\_DIR/jars/" echo "🚀 Running Maven Central compatibility check..." python3 check-legacy-libs.py "$OUTPUT\_DIR/jars"

**✅ 2. Python Script:**check-legacy-libs.py

python

CopyEdit

import os import sys import zipfile import requests from collections import defaultdict deprecated\_keywords = { 'spring-core': 'org.springframework:spring-core', 'spring-context': 'org.springframework:spring-context', 'spring-web': 'org.springframework:spring-web', 'commons-logging': 'commons-logging:commons-logging', 'commons-lang': 'commons-lang:commons-lang', 'jackson-databind': 'com.fasterxml.jackson.core:jackson-databind', 'jaxb-api': 'javax.xml.bind:jaxb-api', } def identify\_lib(jar\_file): try: with zipfile.ZipFile(jar\_file, 'r') as zipf: for name in zipf.namelist(): if 'META-INF/maven/' in name and name.endswith('pom.properties'): with zipf.open(name) as pomfile: lines = pomfile.read().decode().splitlines() group, artifact, version = "", "", "" for line in lines: if line.startswith("groupId="): group = line.split("=")[1] if line.startswith("artifactId="): artifact = line.split("=")[1] if line.startswith("version="): version = line.split("=")[1] if group and artifact: return f"{group}:{artifact}", version except: return None, None return None, None def suggest\_version(group\_artifact): url = f"https://search.maven.org/solrsearch/select?q=g:{group\_artifact.split(':')[0]}+AND+a:{group\_artifact.split(':')[1]}&rows=1&wt=json&core=gav&sort=version+desc" resp = requests.get(url) if resp.status\_code == 200: docs = resp.json().get("response", {}).get("docs", []) if docs: return docs[0]['v'] return "?" def main(jar\_dir): all\_jars = [os.path.join(jar\_dir, f) for f in os.listdir(jar\_dir) if f.endswith('.jar')] report = defaultdict(dict) for jar in all\_jars: ga, version = identify\_lib(jar) if not ga: continue for keyword, full\_ga in deprecated\_keywords.items(): if keyword in ga or keyword in jar: latest = suggest\_version(full\_ga) report[ga]['found'] = version report[ga]['suggested'] = latest print("\n📋 Compatibility Report:") print("=" \* 60) for lib, info in report.items(): print(f"{lib}") print(f" ⚠️ Found: {info['found']}") print(f" ✅ Suggested: {info['suggested']}\n") if \_\_name\_\_ == "\_\_main\_\_": if len(sys.argv) != 2: print("Usage: python3 check-legacy-libs.py <jar-dir>") sys.exit(1) main(sys.argv[1])

**✅ Example Output**

yaml

CopyEdit

📋 Compatibility Report:

============================================================

org.springframework:spring-core

⚠️ Found: 4.3.22.RELEASE

✅ Suggested: 5.3.32

commons-logging:commons-logging

⚠️ Found: 1.1.1

✅ Suggested: 1.2

com.fasterxml.jackson.core:jackson-databind

⚠️ Found: 2.4.0

✅ Suggested: 2.17.0

**🧠 Benefits**

* Quickly find **legacy or deprecated** libraries.
* Get **latest compatible** versions.
* Helps you **plan upgrades** for Java 11 & JBoss EAP 8.

Would you like:

* A ZIP of the full tool?
* A Maven plugin version of this checker?
* Spring Boot dependency scanner support (via spring-boot:dependencies)?

Let me know and I’ll prep it!