# Service Code Changes - Java



Document Level Classification

300



# Introduction

When migrating your application from OldProd to Substrate, code changes will need to be made to your application for compatibility with the platform.  This article highlights the necessary code changes to port your Java application from OldProd to Substrate.

# Logging (Required)

Changes here are concerned with ensuring logs are emitted to standard out for collection by the log collector agent in Kubernetes. The approach is different depending on the underlying framework. The current logging configuration should be the default with the Kubernetes support enabled via an Environment Variable (**ENABLE_STDOUT_LOGS**) set to **true**. Set this environment variable to true in your Helm values.yaml

## Jetty

### 1. Add the logstash encoder dependencies to build.grade

```
sharedDependencies.logback.classic
sharedDependencies.logstashEncoder
```

### 2. Add Appender to logback-overrides.xml

```
    <if condition='!isDefined("log.app.noStdout") &amp;&amp; !isDefined
      <then>
        <appender name="APP_STDOUT" class="ch.qos.logback.core.ConsoleA
          <encoder class="net.logstash.logback.encoder.LogstashEncoder"
            <includeMdcKeyName>EPIC_SRC_IP_ADDRESS</includeMdcKeyName>
            <includeMdcKeyName>X-Epic-Correlation-ID</includeMdcKeyName
            <includeMdcKeyName>EPIC_USER_ID</includeMdcKeyName>
            <includeMdcKeyName>EPIC_CLIENT_ID</includeMdcKeyName>
            <includeMdcKeyName>EPIC_SESSION_ID</includeMdcKeyName>
            <mdcKeyFieldName>EPIC_SRC_IP_ADDRESS=IP</mdcKeyFieldName>
            <mdcKeyFieldName>X-Epic-Correlation-ID=CORRID</mdcKeyFieldN
            <mdcKeyFieldName>EPIC_USER_ID=USER</mdcKeyFieldName>
            <mdcKeyFieldName>EPIC_CLIENT_ID=CLIENT</mdcKeyFieldName>
            <mdcKeyFieldName>EPIC_SESSION_ID=OAUTHTOKEN</mdcKeyFieldNam
          </encoder>
        </appender>
```

```
      </then>
    </if>
```

### 3. Set ENABLE_STDOUT_LOGS to true in Helm Values

```
epic-app:
  containers:
    my-service:
      image:
        name: hub.ol.epicgames.net/epicgames/my-service
      environment:
        ENABLE_STDOUT_LOGS: true
```

## Spring

### 1. Add CompressedStackTraceConverter.java

```
/*
 * Copyright Epic Games, Inc. All Rights Reserved.
 */
package com.epicgames.fortnite.core.util.logging;

import ch.qos.logback.classic.pattern.ThrowableProxyConverter;
import ch.qos.logback.classic.spi.ILoggingEvent;
import ch.qos.logback.classic.spi.IThrowableProxy;

public class CompressedStackTraceConverter extends ThrowableProxyConver
  private static final String LINE_SEPARATOR = System.getProperty("line
  private static final String STACK_TRACE_LINE_SEPARATOR = " | ";

  @Override
  public String convert(ILoggingEvent event) {
    final IThrowableProxy tp = event.getThrowableProxy();
    if (tp != null) {
      final String original = super.convert(event);
```

```
      return original.replaceAll(LINE_SEPARATOR, STACK_TRACE_LINE_SEPAR
    } else {
      return "";
    }
  }
}
```

**2. Add Appender to logback-overrides.xml**

```
<if condition='!isDefined("log.app.noStdout")'>
                <then>
                        <!-- Override property in logback.xml in epic-c
                        <conversionRule conversionWord="ex" converterCl

                        <property name="sentry.tag_list" value="EPIC_PA

            <property name="LOGGING_PATTERN" value="${log.app.pattern:-

            <appender name="APP_STDOUT" class="ch.qos.logback.core.Cons
                <encoder>
                    <pattern>APPLICATION | ${LOGGING_PATTERN}</pattern>
                </encoder>
            </appender>

            <appender name="ANALYTICS_STDOUT" class="ch.qos.logback.cor
                <encoder>
                    <pattern>ANALYTICS | ${LOGGING_PATTERN}</pattern>
                </encoder>
            </appender>

            <appender name="ACCESS_STDOUT" class="ch.qos.logback.core.C
                <encoder>
                    <pattern>ACCESS | %msg%n</pattern>
                </encoder>
            </appender>

            <logger name="com.amazonaws.auth.AWSCredentialsProviderChai
```

```
                <appender-ref ref="APP_STDOUT"/>
            </logger>

            <logger name="ANALYTICS_EVENT_LOGGER" level="INFO" additivi
                <appender-ref ref="ANALYTICS_STDOUT"/>
            </logger>

            <logger name="EPIC_ACCESS" level="INFO" additivity="false">
                <appender-ref ref="ACCESS_STDOUT"/>
            </logger>

            <root level="${log.app.level:-INFO}">

                <appender-ref ref="APP_STDOUT"/>

                <!-- Sentry -->
                <if condition='isDefined("sentry.dsn")'>
                    <then>
                        <appender-ref ref="SENTRY"/>
                    </then>
                </if>

                <!-- New Relic -->
                <appender-ref ref="NEWRELIC_NOTICE_ERROR"/>
            </root>
                </then>
        </if>
```

### 3. Set ENABLE_STDOUT_LOGS to true in Helm Values

```
epic-app:
  containers:
    my-service:
      image:
        name: hub.ol.epicgames.net/epicgames/my-service
```

```
      environment:
        ENABLE_STDOUT_LOGS: true
```

**4. Add support for ENABLE_STDOUT_LOGS to entrypoint-start.sh**

```
# if ENABLE_STDOUT_LOGS == false or missing, execute this condition
if [ -z "$ENABLE_STDOUT_LOGS" ]; then
   EPIC_JAVA_OPTIONS+=(
      -Dlog.app.noStdout=true
      -Dlog.capturestd=true
   )
fi
```

**5. Remove -Dlog.capturestd=true from EPIC_BASE_JAVA_OPTIONS in entrypoint-start.sh**

```
EPIC_BASE_JAVA_OPTIONS=(
    -Dintent=$EPIC_INTENT

    # Specify default ports for deployed environment
    -Dcom.epicgames.app.services.admin.port=8080
    -Dcom.epicgames.app.services.public.port=9080

    -Dlog.app.level=${APP_LOG_LEVEL:=INFO}
    -Dlog.directory=$APP_LOG_DIR

        # REMOVE THIS OPTION
        #-Dlog.capturestd=true

    # If there is a DSN override value, use that;
    # otherwise, use the DSN value commonly found in the ansible tag_ro
    -Dsentry.dsn=${SENTRY_DSN_OVERRIDE:-$SENTRY_DSN}
    -Dsentry.environment=$EPIC_INTENT

    # In the future the bridge IP should be changed to use a container
    # to a colocated rsyslog container scheduled on the same host. By d
```

```
    # the Docker daemon will use 172.17.42.1 as the bridge IP address.
    -Dlog.syslog.host=$DOCKER_BRIDGE_IP
)
```

# SIGTERM Handling

Ensure any custom entrypoint.sh script supports SIGTERM handling to ensure smooth pod deregistration during deployments and scaling:

**Insert at the end of custom entrypoint script**

```
echo "catching SIGTERM"
/epicgames/bin/entrypoint.sh "$@" &
pid=$!
trap 'kill -SIGTERM $pid; wait $pid' SIGTERM
wait $pid
```

# EC2 Metadata

Some services depend on EC2 metadata by querying the underlying nodes metadata endpoint for some data such as the nodes IP address or Hostname. Since this cannot be done in Kubernetes pods, you will need to adapt your service code to support both EC2 metadata and Kubernetes Pod IP lookups through a service configuration value. If you already have a configuration value you can set it in the containers environment variables like so:

```
epic-app:
  containers:
    service:
      environmentValueFrom:
```

```
        config.com.epicgames.someValue.hostname_override:
          fieldRef: status.podIP
```

Other values available:

| fieldRef | Example Value |
| --- | --- |
| `spec.nodeName` | ip-10-104-112-83.ec2.internal |
| `metadata.name` | my-service |
| `metadata.namespace` | prod |
| `status.podIP` | 10.104.65.254 |
| `spec.serviceAccountName` | default |

Kubernetes Docs: https://kubernetes.io/docs/tasks/inject-data-application/environment-variable-expose-pod-information/#use-pod-fields-as-values-for-environment-variables

# IRSA Credential Providers

In some cases you may need to adapt your services' AWS credential chain to support IRSA.

# Found something that needs to be here?

Let us know here: #cloud-governance-oldprod-exodus-support-ext