A technical guide for monitoring Adobe® LiveCycle® ES deployments

Table of contents

This technical guide provides insight and recommendations for monitoring Adobe LiveCycle ES (Enterprise Suite) Update 1 in production. This information is specifically targeted at a LiveCycle ES Update 1 or later deployments, as prior versions did not have the monitoring features outlined in this guide.

This document is formatted into three logical sections.

- "Section 1: LiveCycle ES system monitoring" covers an overview of monitoring LiveCycle ES from a complete system perspective.
- "Section 2: Internal LiveCycle ES monitoring" looks at specific internal monitoring capabilities inside LiveCycle ES.
- "Section 3: Monitoring LiveCycle ES" outlines the practical aspects of monitoring a real-world LiveCycle ES deployment.

Note: This technical guide is specifically written to cover the Foundation-based LiveCycle ES solution components because of their more complex dependencies and requirements. This guide covers all the solution components except LiveCycle Data Services ES.

Section 1: LiveCycle ES system monitoring

It is helpful to understand the high-level deployment of Adobe LiveCycle ES at the system level. This holistic view is critical because LiveCycle ES interacts with several infrastructure components in the deployment that directly impact its operation. Depending on the level of monitoring required, you may decide to focus on some aspects more then others. Also, monitoring may be more useful at different stages of the project. Tuning and stress testing may benefit from monitoring during the setup of the environment, while runtime monitoring may be more important to other projects.

A LiveCycle ES deployment contains the following infrastructure of interest to monitoring:

- Operating systems
- Databases
- Java Virtual Machines (JVMs)
- Java application servers
- LiveCycle ES servers
- Shared directories
- Native processes
- LDAP directories
- Network and network devices
- E-mail servers

Adobe

Monitoring this infrastructure can be done for several reasons. System availability for Service Level Agreements, performance tuning, and failure analysis are common examples. Enterprise infrastructure—such as databases, operating systems, and network devices—has standard built-in monitoring capabilities provided by the vendors that supply them. In addition, specialist management tools such as HP OpenView and IBM Tivoli Monitoring can be used to collate the different monitoring information provided by the different components in the system into a single actionable view. These management tools are extremely valuable when there are multiple components to monitor and they can also add higher level events such as personnel notification (call administrator via pagers) and active issue resolution (recycle a system in a cluster).

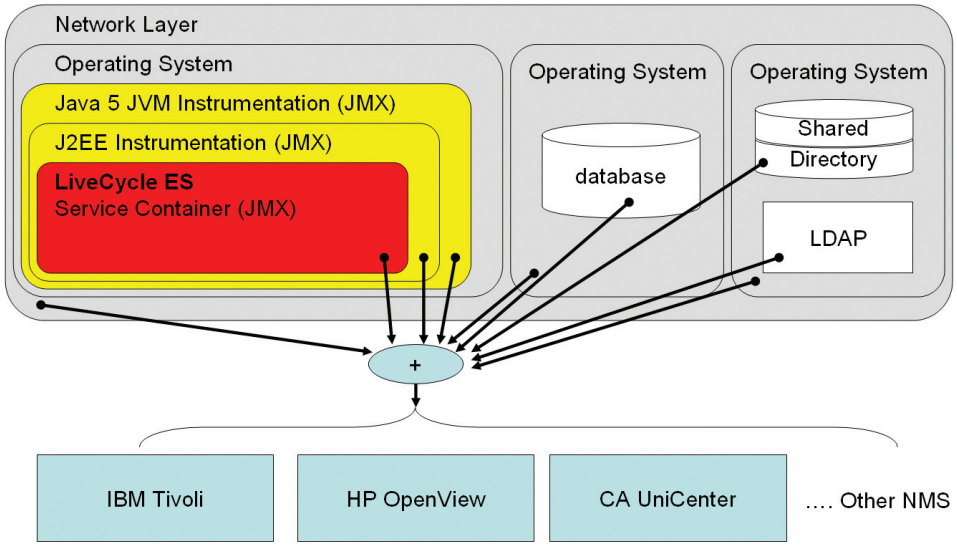

Figure 1. Overview system monitoring

This guide will not go into detail about how to monitor the non-Java components of the LiveCycle ES deployment. The non-Java components are standardized, and there are commonly available solutions to monitor them. The Java monitoring options are specialized enough to need more review for people familiar with other enterprise monitoring systems.

Java monitoring

The Java monitoring and instrumentation standard is known as Java Management eXtensions (JMX) JSR 160. JMX provides a monitoring and management framework that is included in the application servers and JVMs supported by LiveCycle ES. There are generally two levels of JMX support in J2EE: standard and custom. LiveCycle ES version 8.2 provides custom-level JMX support, which means that mappings need to be made between the LiveCycle ES JMX instrumentation and the monitoring system. Infrastructure that has standard JMX support will have predefined mappings for the most common monitoring systems.

Real-world JMX implementations provide a hierarchical information model that allows for detailed runtime analysis of Java-based systems. JMX monitoring does not significantly alter runtime performance of the instrumented system.

Generally, JMX monitoring provides runtime information such as notifications, statistics, and state. LiveCycle ES does not support notifications because notification at the monitoring system level is the most flexible option for real-world use cases. Monitoring agents are best used to amalgamate the information conveyed over JMX statistics and state and make the necessary alerts using their richer communication (pagers, e-mail, SMS, and so on) and resolution options.

Management interfaces and statistics are implemented as MBeans. In a JMX environment, the MBean Server provides runtime support for MBeans and thus offers security, access, and protocol support. The MBeans can be looked up and used from within the JVM directly. Thus, a web application can be created to publish the MBean information, for example. More commonly, the MBeans will be accessed remotely. The out-of-the-box MBean Server protocols typically include Remote Method Invocation (RMI) and HTTP. There are several tools available that will connect directly to the MBean Server RMI interfaces, and this is a good option for simple monitoring requirements. (See Appendix.)

Conspicuously missing is SNMP, which is the common protocol for systems management. SNMP is available as an add-on for the common application servers. Typically when a management system is being used, such as OpenView or Tivoli, there will be adapters available with the system. Adapters translate the protocol to one more suited to the management system in use.

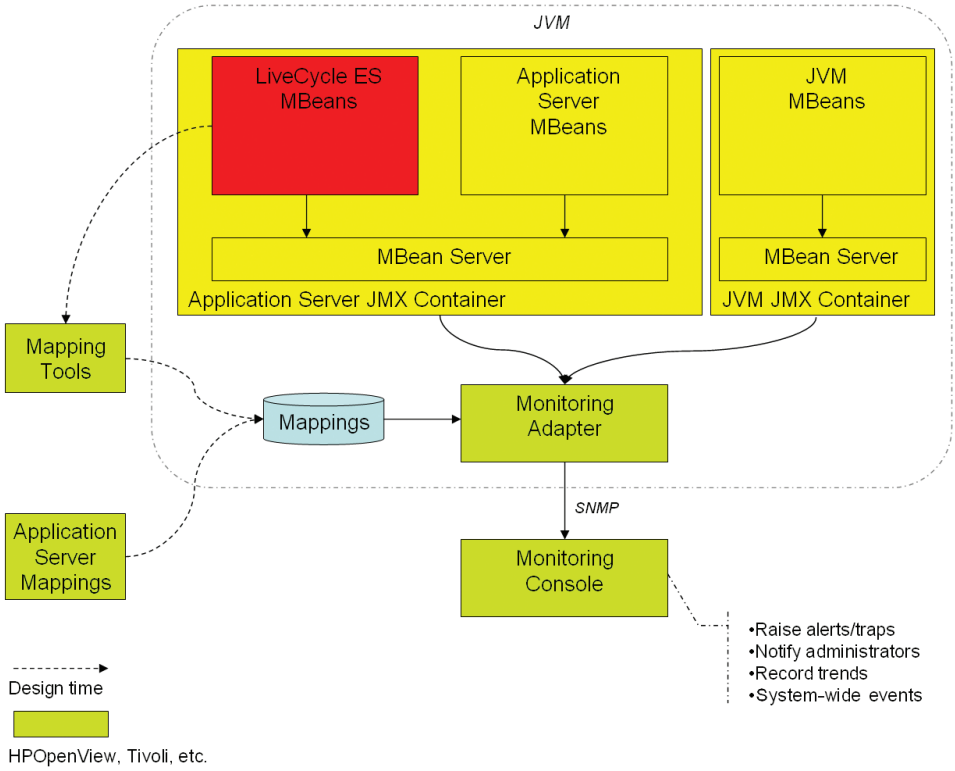

Figure 2. JMX monitoring

Figure 2 is an example of a JMX monitoring topology. A third-party management system has been used with an adapter to implement communications between the application server and the monitoring console. The adapter uses customer mappings, built with tools provided with the monitoring system, to read the LiveCycle ES instrumentation. Because the monitoring system had prebuilt mappings for the application server and JVM, these mappings were used to get at instrumentation in those locations. The monitoring adapter polls the MBean Server for statistics. The management console either polls the adapter or receives events from the adapter. Administrative events and thresholds of interest can be set in the monitoring console and connected to pager messages to notify administrators of issues. In addition, the monitoring console can record trends and historical data for future analysis. This is important because JMX keeps all statistics in memory, and they are lost when the system is recycled.

Section 2: Internal LiveCycle ES monitoring

The LiveCycle ES Update 1 service container is instrumented with public MBeans for monitoring use cases. There are other MBeans in LiveCycle ES, but they are not supported for use by customers. The LiveCycle ES service container is responsible for executing all LiveCycle ES services; thus, any request to a LiveCycle ES service will be instrumented. LiveCycle ES services include Document Service Container services and LiveCycle ES processes. At this time, there is no instrumentation outside the LiveCycle ES service container. This is not a significant limitation because the service container is the central processing engine for all LiveCycle ES transactions. Authentication failures are one example of events that are not visible with the current implementation, because authentication happens before the service container receives the request.

MBeans are only created for services and processes that are activated (available to respond to requests). The MBean instances for each service are created when the LiveCycle ES service container starts or when new services and processes are created.

Note: The information in the rest of section 2 is pre-release and may be subject to change prior to the release of LiveCycle ES Update 1.

There are two registered LiveCycle ES MBeans created for each activated service. They provide navigation and statistic information.

- ServiceStatistic—This MBean provides information about Service name and its version.
- OperationStatistic—This MBean provides information about statistics of LiveCycle ES services. Relevant runtime statistics about a particular service such as invocation time, number of errors, and so on are encapsulated in this MBean.

OperationStatistic statistics are MBean's attributes and can be navigated with the hierarchy tree:

Adobe Domain Name: This depends on the Apps Server. If the Apps Server does not have a domain, it defaults to adobe.com.

ServiceType: AdobeService is the name we use to list all services.

AdobeServiceName: This is the Service Name or Service ID.

Version: This is the version of the service.

Operation Statistics:

- Invocation time: This refers to the time taken for the execution of the method. This is the time it takes to go through validation, transaction, authorization, and invocation. This does not include the time the request is serialized, transferred from client to server, and de-serialized.
- Invocation count: This is the number of times the service has been invoked since the application server started.
- Average invocation time: This is the average time for the execution of the method.
- Max invocation time: This is the longest invocation time.
- Min invocation time: This is the shortest invocation time.
- Exception count: This is the number of invocations that failed in the service container to the method.
- Exception message: This is the error message from the last failure.
- Last sampling date time: This is the date of the last invocation.
- Time unit: This refers to the units of time being used.

Section 3: Monitoring LiveCycle ES

There are many reasons to monitor a LiveCycle ES deployment. Outlined in this section are some things you may consider.

System Tuning:

It can be very important during the system deployment and tuning phase to do stress testing of the environment to verify the operational envelope. Doing so often requires an understanding of which subsystems are saturating first and altering configurations to ascertain the effect.

- Monitor heap usage and garbage collection activity, especially on large 64-bit, multi-CPU systems.
- Tune caches and resource pools.
- Monitor CPU and disk utilization on the LiveCycle ES server and on the database server.

Small deployment:

Performance is often not the primary concern in smaller deployments. But the step in complexity when moving from a small deployment to a multiserver (cluster) deployment can be significant. Thus, it can be advantageous to expand the boundaries of a single server deployment to avoid the added complexity of multiserver deployments. Also, intermittent or longevity failures can be hard to track with log files alone. In both these cases, using monitoring software to directly monitor the application server and LiveCycle ES deployment can be very informative. All that may be required is using JConsole (appendix) to monitor very specific statistics over time and with load.

Mission-critical system:

Continuous runtime monitoring is recommended if the LiveCycle ES deployment is part of a mission-critical system that requires very high availability and reliability. All critical dependencies are candidates for monitoring.

- Application server connection pool utilization
- JVM heap space utilization and garbage collection
- Failures for any LiveCycle ES services being used in critical applications
- Database CPU utilization
- Database table space utilization
- Shared Directories, specifically the Global Document Storage Directory (GDS)

Appendix

JMX Consoles for direct monitoring of the JMX server in isolation

See Sun Developer Network article, "Using JConsole to Monitor Applications," (*http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html*) by Mandy Chung, December 2004.

JConsole comes with Java 5 and later versions and is a Swing-based desktop application.

Technical guide feedback

We welcome your comments. Please send any feedback on this technical guide to *LCES-Feedback@adobe.com*.

Adobe

Adobe Systems Incorporated
345 Park Avenue
San Jose, CA 95110-2704
USA
www.adobe.com

95010483 03/08