



White Paper

# Application-Oriented Monitoring in a Cloud Component World

**Author:** Tom Lubinski

**Publication Date:** February 2011

**Abstract:**

Answering the question, “how is my critical software application really performing,” has become increasingly difficult over the last decade. Reliance on componentization and service-oriented architecture has simplified application development, but has correspondingly increased the complexity and the number of dependencies within the application environment, greatly complicating the health-state monitoring of critical applications.

The cloud revolution promises huge benefits in IT savings, but initially introduces even more complexity with new integration points and interfaces. This paper discusses the challenges inherent in monitoring complex applications based on component architectures, and how, if properly monitored, the nature of the cloud itself may ultimately provide the solution.

---

## Copyright

© 2011 Sherrill-Lubinski Corporation. All Rights Reserved.

### Trademarks

SL Corporation, SL-GMS, RTView and the SL logo are trademarks or registered trademarks of Sherrill-Lubinski Corporation in the United States and other countries.

---

## Contents

Application-Oriented Monitoring .....	4
The Challenges of Monitoring Component-Oriented Applications .....	5
The Cloud Revolution Advances Componentization.....	6
Application Health State and Data Flows in Real-time.....	7
Automate Monitoring From Configuration Data – the Holy Grail .....	10
About SL Corporation .....	12

---

## Application-Oriented Monitoring

Large companies today are extremely aware of the importance of maintaining availability and performance of critical applications on which the entire operation may be dependent. So much work is automated and/or performed on-line that a single brief outage can result in millions of dollars in lost revenue.

The applications that provide such value in large enterprises are typically quite complex, consisting of many independent service components passing data back and forth using messaging systems, application and web service technologies, or simple file transfers. Reliance of these applications on shared subsystems has grown significantly in the last decade, making it increasingly difficult to understand how any one application is performing when the underlying components are shared.

The challenge is so great that few companies are able to solve it using in-house development resources. This has resulted in the proliferation of commercial products offering Application Performance Monitoring (APM) solutions attempting to address this problem. However, many of these solutions are vendor-specific, limited to monitoring of individual components, or address only a single aspect such as network traffic profile.

The author of this article and SL Corporation have over 25 years of experience in real-time monitoring and visualization applications, with particular expertise in Java. The company's RTView product has been specially developed to handle extensive requirements around performance monitoring of component-based applications, with a particular emphasis on understanding, visualizing and managing the health state of critical applications that rely on shared services.

Application support groups need to monitor their own applications; they don't care about other applications that share the same component services. As such, bringing together information from many subsystems, and yet presenting it in an application-oriented fashion, is critical to effective monitoring.

---

Complicating matters further, there is much talk about the promise of deploying applications into the “cloud” with a goal of reducing costs related to IT management. While this may eventually come about, big changes take time. The complex nature of these applications means they are likely to be migrated in stages. This means there will be even more integration points that must be monitored – places where the transfer of data from legacy components to and from new cloud-based modules can fail.

## **The Challenges of Monitoring Component-Oriented Applications**

Over the last decade, the challenge of application monitoring has worsened as applications have grown in sophistication and complexity, and have become increasingly component-oriented. The productivity of application developers has been greatly enhanced by the use of shared service components such as application servers, enterprise beans, messaging middleware and, more recently, distributed caching systems. However, at the same time, these greater dependencies make it increasingly difficult to understand how an application is really performing.

Messaging middleware provides a good example. Oracle Fusion Middleware, TIBCO EMS, and IBM MQ are used by many organizations to reliably transmit important data between application subsystems. Dozens of disparate applications might use the same message broker to transmit data — if any one of them is overloading the broker, they can all be impacted. It is not easy to determine who the culprit is and who may be negatively affected.

Worse yet, since resources are shared, one application having trouble may not trigger an alert on the shared resource. Often, the first indication that there is a problem is when screaming users call the support team and tell them their application is taking two minutes to complete a job that normally takes two seconds.

Since no one knows which applications use which resources, the result is the all-too-familiar “war room” where all hands are called together in the middle of the night to figure out what went wrong. The organization may have every major monitoring tool, but most are siloed or point solutions showing data about a single component. Everyone scrambles, trying to make sense of all the data, until they finally find the source of the problem, sometimes hours later.

---

RTView was designed specifically to eliminate this frantic search for root cause data by presenting health-state information collected from component subsystems in a centralized, organized and application-oriented fashion.

But before going there, a bit more about the cloud...

## **The Cloud Revolution Advances Componentization**

All the talk about the “cloud revolution” can be confusing. The term means different things to different people. Some scoff at the suggestion that the cloud is something new, arguing that companies such as Amazon, eBay and others have been “in the cloud” since the 90s.

Others recognize that new technologies have enabled the creation of “virtual” servers, making it possible to replicate a system configuration hundreds of times over. There is no hardware to move around, only software. This means that deployment can be accomplished orders-of-magnitude faster than in previous generations. It is equivalent to going from the hand-wiring of circuit boards to the use of a programming language to control logic configuration in software.

Some visionaries, like Jerry Cuomo, Chief Technology Officer of the IBM WebSphere Division, recognized this sea of change years ago, and have actively promoted cloud-based solutions within their organizations. In a 2008 InfoQ interview, Jerry describes “Atomic Virtualization” and “Molecular Virtualization” as techniques for “freeze-drying” a server configuration and information that can be used to pre-install and configure that server so that it can be hot deployed. He says “where it really starts to get cool is when you can create collections of these things... this allows you to freeze dry actual server topology, a group of servers.” The interview can be found at <http://www.infoq.com/interviews/jerry-cuomo-cloud-computing>.

The introduction of private-cloud and public-cloud computing is a natural consequence of making shared compute resources available via these virtualization techniques. It puts huge amounts of computing power into the hands of the masses of business users who just don't have the time or resources to manage the hardware necessary to support large-scale deployments. It truly is a revolution.

---

Applications can now be completely decoupled from underlying host hardware. What used to be considered immutable infrastructure, the hardware servers are now configurable components as well. One cannot assume that an application is running on a specific host or network address.

Whether the environment is a private cloud or a public one, the problem of monitoring the health state of an application now becomes even more complicated. The entire deployment architecture is componentized, introducing yet another level of indirection that must be accounted for. Everything upon which an application is built, from the servers up through the entire middleware stack, must be viewed and monitored as components.

In the short term, applications will not be transported in their entirety to the cloud. Parts of the application will remain in legacy systems. A lot of new integration points will need to be monitored. While standardization and componentization promise a utopian future application landscape, the short term reality is much different. The monitoring challenges discussed previously may only be getting worse.

## **Application Health State and Data Flows in Real-time**

Most applications, implemented in the cloud or not, suffer from a common problem: lack of real-time visibility into their true health state and performance levels.

Traditional systems management tools that concentrate on infrastructure metrics such as CPU and memory usage are usually found to be inadequate in a shared component world. The siloed or point solutions from component vendors offer insight into how a particular subsystem is performing, but users are left with the daunting task of associating subsystem metrics with the many applications that are dependent on it in order to identify a performance-hogging culprit.

Other technologies, such as network packet analysis or method-level VM instrumentation, can help identify hot spots within an application's code base or interface to the outside world. However, problems often result from the failure of data flow from one component to another when a required feed process is down or when a file cannot be found. Problems of this type are difficult to identify with such tools.

---

True visibility into an application's health state requires a comprehensive and integrated view of the state of multiple components on which it is dependent. Even more important is a filtering mechanism that permits users to quantify an individual application's usage of shared resources.

Some APM solutions have emerged to tackle this problem head-on. SL Corporation, with its RTView product, offers many APM features for real-time monitoring of application health state and historical baseline analysis. Several features are especially important when monitoring applications based on multiple component subsystems, such as the ability to:

- 1) Collect monitoring data from all relevant sources and make it available in real-time to the people who need to see it. This requires a modular and scalable monitoring system, with multiple instances deployed across regions or subnets, providing summary information as well as drilldown to component details.

This part of the solution is especially important during critical outages, or when users are complaining about slow performance, and it is imperative to quickly determine which subsystem is at fault. Having all data from every component "at your fingertips" is critical to solving this problem.

- 2) Present monitoring data in "contextual" views that show in a single display the flow of data through all components relevant to a specific application. Users think of these as Visio diagrams "coming alive" with real-time data flows showing throughput metrics and the health state or availability of each component. Users need to be able to construct these diagrams in a custom fashion and associate objects in the diagram with specific metrics.

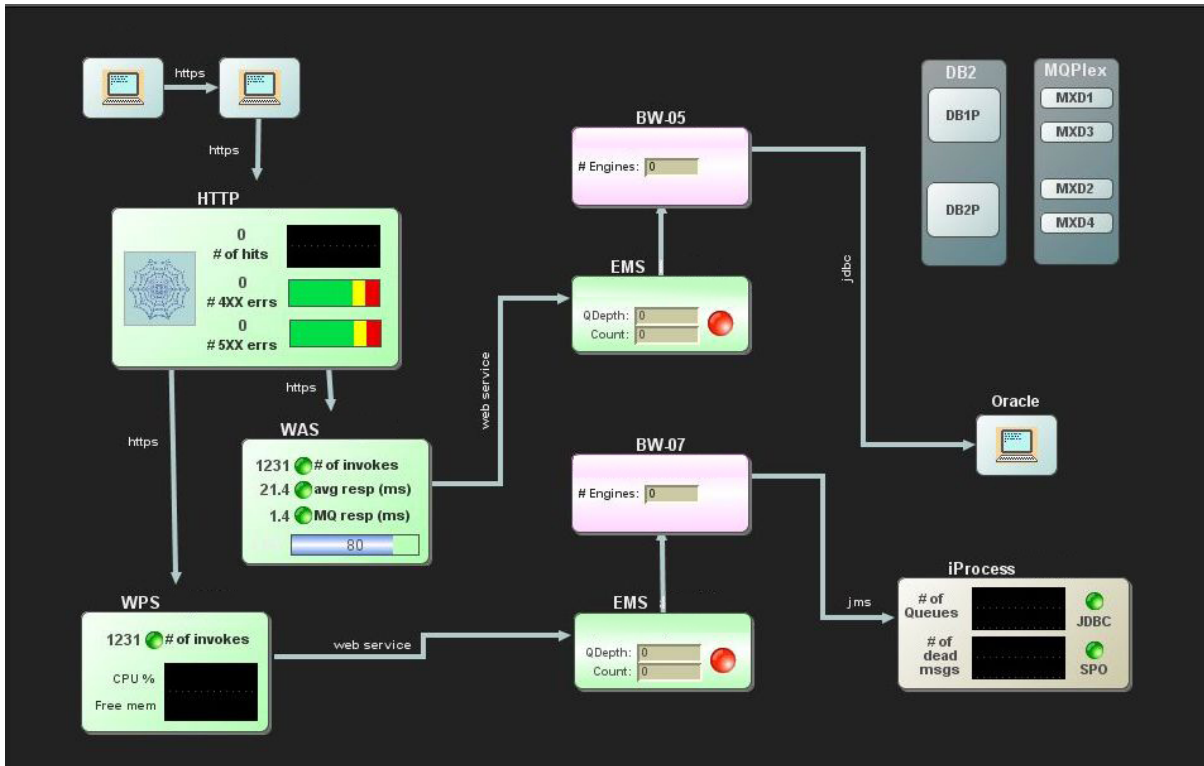


Figure 1 - Application Data Flow Display showing metrics from several components

This capability is important because application support people often think of their applications in terms of data feeds and services that pass data to one another. Each component must be up and running, with data flowing between them, in order for the application to be performing properly.

3) Filter monitoring data by application so that it is possible to quickly determine which applications are affected when a problem occurs in a shared component. This capability is critical in messaging systems, for example, where a single server can be shared by dozens of applications and each application uses only a few specific topics or queues. Most users only want to see the metrics that are relevant to their applications.

These techniques define some important characteristics required for a comprehensive application-oriented monitoring solution. RTView has been highly successful in its ability to provide powerful application-oriented views to numerous organizations. When monitoring multiple and/or extremely large-scale applications, it is crucial to be able to automate as much of the monitoring displays and information as much as possible.

## Automate Monitoring From Configuration Data – the Holy Grail

Many organizations maintain a database of information about their hardware and software inventory. It seems logical that such information could be used to automatically populate a series of displays showing the health state of applications that make use of that inventory.

In order to support this type of automation, the monitoring system must be flexible enough to have its major features be data-driven. RTView is an example of a system that is highly data-driven in that every aspect of its configuration can be driven by data tables that describe data sources, display styles, layouts, object relationships and much more. With this degree of flexibility, RTView can be deployed and its data collection parameters automatically configured from data maintained in the inventory database. Likewise, dependency information can be presented automatically in monitoring displays showing important application/component relationships as well as real-time metrics, all without custom development.

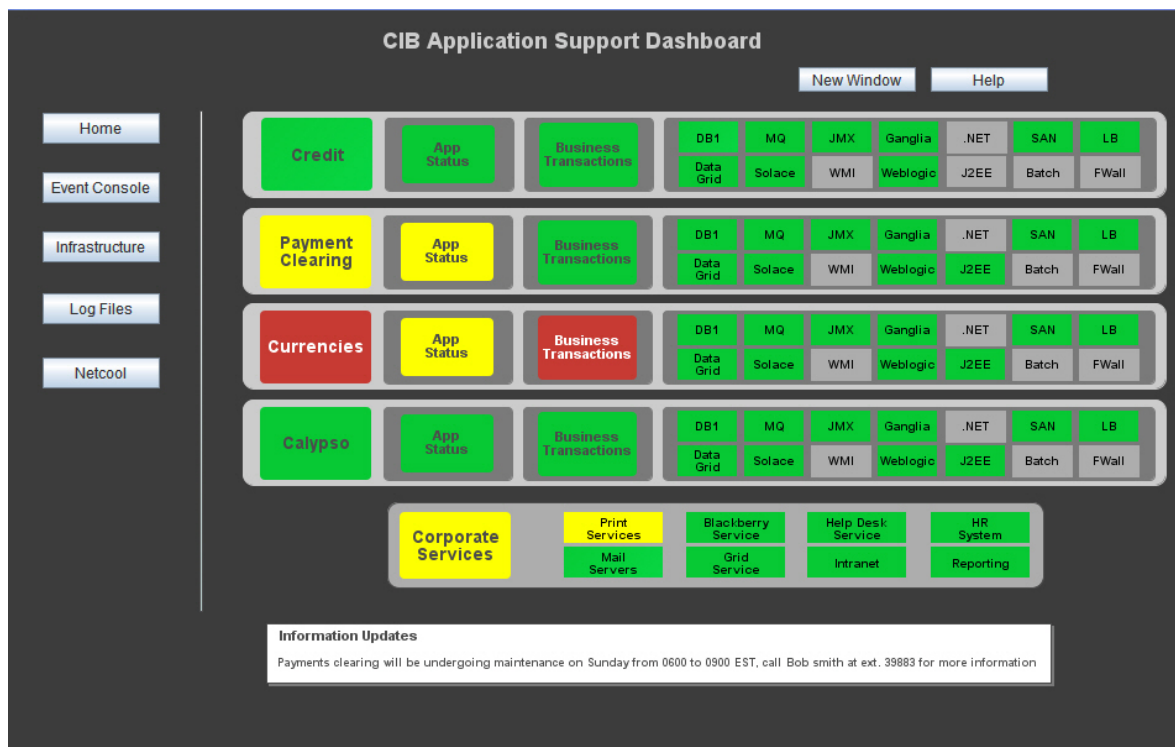


Figure 2 – Application Dependency Display showing components relevant to each application

---

There is a broad range in the types of configuration information that users have access to. Some keep a simple spreadsheet of application dependencies on middleware components or servers. Others maintain a much more sophisticated “configuration management database,” or CMDB. As long as the data in the CMDB accurately represents the current state of the system, the data-driven configuration and monitoring displays provide a useful view of the true health state of the system.

However, it can be difficult for organizations to manually maintain a database of configuration information when servers are constantly being replaced, reassigned, etc. A manually managed database can become out-of-date quickly, and the monitoring system based on it becomes unreliable.

For this reason, many organizations have sought to take advantage of tools that are designed to “autodiscover” the topology of a network and the dependency of the applications on various software components or subsystems. While this is a great idea, in practice, the discovery mechanism has shortcomings and needs to be constantly updated in order to stay on top of new components or quirks in the configuration.

RTView has been successfully used by several customers in automatically creating monitoring views from information that is obtained from an auto-discovery tool. In most cases, it has been necessary to introduce a manual review step before committing the model and making the process automatic, due to the large number of exceptions.

When applications are deployed into a cloud environment, all information about infrastructure, middleware and applications themselves is contained in a configuration database. Since the underlying components are all virtualized and deployed, all the references are also parameterized. For example, all the information about network addresses, names of servers, names of deployed app servers and message systems must be contained in this database.

In this scenario, there is no need to “discover” this information. The dependency information is already contained in the deployment database. A natural evolution of this is that monitoring will no longer be viewed as an add-on or an afterthought. Since deployment of all subsystems will be automated with cloud provisioning, even the monitoring applications can be deployed and configured automatically. RTView is rapidly emerging as the monitoring system of choice in these environments.

---

## About SL Corporation

Over the past 25 years, SL Corporation has become the most knowledgeable and responsive provider of real-time monitoring, analytics and visibility solutions. SL's flagship product, RTView, addresses a broad spectrum of operational visibility challenges spanning end-to-end application performance management (APM), business activity monitoring (BAM) and component-level infrastructure monitoring. RTView also has become the de facto standard for extending the visualization of complex event processing (CEP) engines, TIBCO messaging middleware, Oracle Coherence data grids and custom applications. SL's exclusive focus on real-time visibility solutions, commitment to customer success and partner-centric culture are why thousands of industry leaders have chosen to work with SL to support their most critical applications and businesses. SL Corporation can be reached at +1 415-927-8400 or on the web at [www.sl.com](http://www.sl.com).



### **Contact Information**

SL Corporation  
240 Tamal Vista Blvd.  
Corte Madera, CA 94925  
+1 415-927-8400  
info@sl.com

**For more information regarding RTView, please visit:**

**[www.sl.com](http://www.sl.com)**