

# Understanding Active Service Level Management

*August 2001*

by Brad Stone, Chief Technology Officer, Resonate

---

## Introduction

Today, e-business is an integral part of doing business. Most companies across a variety of industries are realizing that e-business applications significantly impact their profitability. For some companies, e-business applications are driving sizable revenue growth through new markets, products, and services. For other companies, e-business applications help reduce operating expenditures through greater productivity. Businesses must improve their e-business application service levels (performance and availability) if they are to protect these profits. However, while sustaining e-business service levels and service level management has become a strategic imperative, improving application performance and availability is NOT easy for the following reasons:

- **Complex infrastructure** - The infrastructure underlying e-business applications is becoming quite complex in order to support variable and sophisticated user demands. Applications are being replicated across multiple servers for higher availability and performance. Adding to the complexity, the presentation, business logic, and data storage components of an application are typically deployed within a tiered architecture across different Web, application, and database servers. Finally, many applications are dispersed geographically to further enhance performance and availability.
- **Organizational and technological silos** - Most enterprise IT groups are organized around specific technologies, platforms, and vendors. For example, an IT department might be divided into a group that focuses on network issues and another that focuses on application development. However, IT groups have now realized that e-business service delivery spans networks, systems, and applications, and in many cases, extends beyond the boundaries of the company to suppliers, partners, and customers. An integrated view of various disparate components underlying an e-business application is critical for effective service level management.
- **End-user perspective** - Most IT organizations are focused on internal metrics such as the availability and utilization of individual servers or network devices. Some more sophisticated organizations may capture service level information for individual silos. In either case, these groups often are not aware of the impact of these problems on end-user service levels. Understanding the impact of IT problems on the end-user experience is critical for prioritizing and resolving problems more quickly. Furthermore, user-centric IT organizations can often find problems before the user calls the HelpDesk.
- **"Drowning" in management data** - Many IT organizations complain about the overwhelming amount of management data (events and/or statistics) that is

collected from various sources. The real issue is what to do with all this data. IT organizations need intelligent management solutions that will help them organize, prioritize, and utilize relevant management data affecting critical e-business applications.

For these reasons, traditional system and network management tools are no longer sufficient for effective service level management. These tools tend to ignore the end-user perspective and focus on specific technologies, platforms, or vendors. These same tools are also typically strong in one or two areas of service level management, such as monitoring and reporting, but lack expertise in other critical areas such as problem diagnostics and resolution (see figure 1 below). These traditional management tools fail to provide an integrated solution that monitors, diagnoses, and resolves problems that impact user service levels.

---

## Active Service Level Management

**So how should IT organizations manage their critical e-business services?**

What's needed is a complete and innovative approach to e-business service level management, called active service level management, that overcomes the limitations of traditional monitoring and reporting solutions.

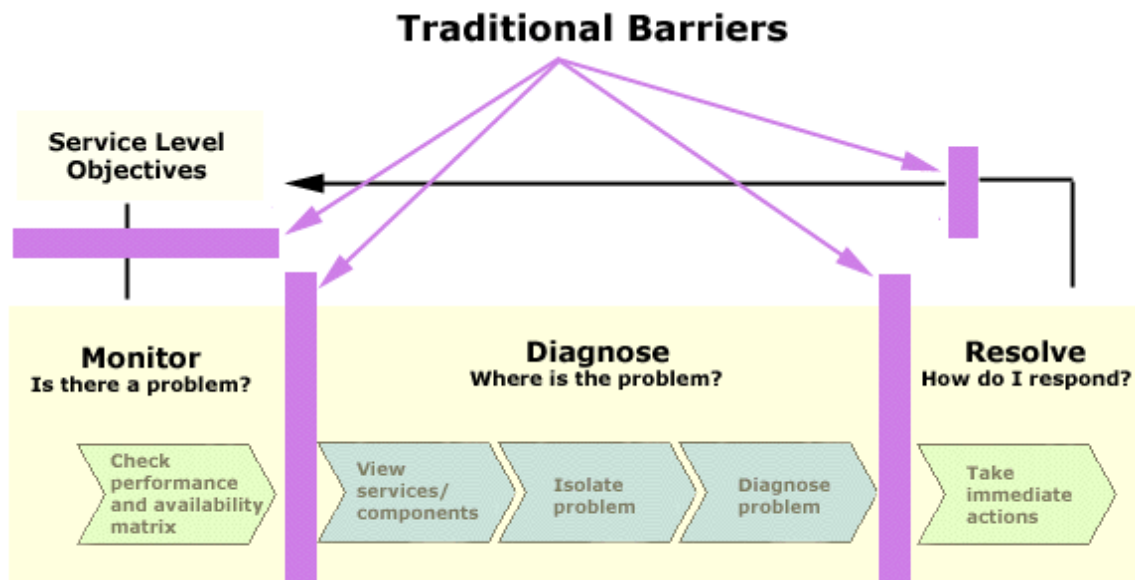


Figure 1. Barriers to Proactive Service Level Assurance

Specifically, the following technical capabilities form the basis for this important e-business management approach:

- **Services Management** - IT groups are realizing that they must manage infrastructures in terms of their ability to deliver e-business services to end-

users. The performance and availability metrics now being measured are generally referred to as service levels, and the goals for those service levels are referred to as service level objectives (SLOs). This important paradigm shift is directly impacting monitoring and performance management tools. It is no longer appropriate to monitor systems, applications, or network devices in isolation and present results in a myopic way.

Furthermore, services management extends beyond monitoring but also impacts the diagnosis and resolution of infrastructure problems.

- **Policy Management** - In addition to services management, it is critical for management tools to provide a complete and integrated solution for resolving critical problems that impact service quality. Active service level management solutions enable IT groups to rapidly and proactively fix e-business infrastructure problems by answering the following questions:
  - Is there a critical problem impacting my end-user service levels?
  - What problems should be addressed first?
  - Where is the exact location of the problem?
  - What additional diagnostics should be performed to better understand the problem?
  - What immediate actions should be taken to either fix the problem or "hide" the problem from the end-user?

While many existing tools attempt to address the first three questions, these same tools do not address the tougher policy questions. Policy Management (i.e., translating events into actions and/or tests) is what truly transforms traditional service level management into active service level management.

- **Distributed Management** - Finally, active Service Level Management solutions help IT groups focus on only the most important or relevant events and/or statistics being gathered from the variety of systems, applications, and network devices underlying e-business services. Most system administrators complain about "drowning" in a sea of management data with their existing tools. They need a solution that distributes the management intelligence of the system closer to the problem. Management solutions that correlate data and take corrective actions locally and automatically enable IT groups to focus on the real issues.

The following sections further describe the core technical capabilities of the active service level management approach for ensuring that e-business services are meeting or exceeding their defined service level objectives.

---

## Services Management

Providing services over the Internet usually implies a set of Web servers initially handling client requests. In turn, these Web servers communicate with applications running on application servers, and finally the application accesses data through a back-end database server. This is generally referred to as a "three-tier model". Along the way the client request may pass through a myriad of devices including firewalls,

load balancers, and routers. All of these various network, system, and application components help to deliver an e-business service.

### Service Model

In order to effectively manage e-business service levels, it is important to manage the inter-dependent components underlying a service. Generally, the first step involves the creation of a Service Model. The Service Model contains a description of the service, its attributes, as well as an ordered list of tiers and elements. A tier represents a collection of similar elements (e.g. Web servers). The elements within a tier are assumed to provide equivalent functionality. Each element has its desired role (e.g. primary, backup) defined as well. The diagram above is a simplified example of what would be included in a Service Model. In this example, the Online Trading service is comprised of four tiers: Firewall, Web Server, Application Server, and Database Server.

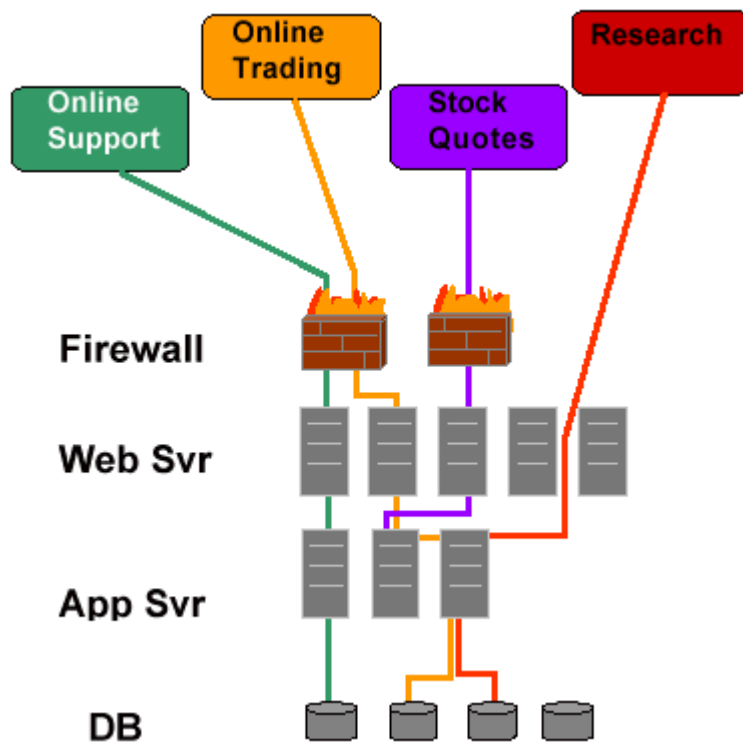


Figure 2. Service Model

Within a tier, the individual servers are listed and the roles of each server are specified. In order to isolate infrastructure problems more quickly, critical performance and availability metrics might also be included in the Service Model for each tier and element. The service model may also list additional dependent services that may affect performance and availability, but are not part of the typical data flow. An example might be a Domain Naming Service (DNS) server or a third-party service such as Stock Quotes.

It should be noted that the administrator decides where a service starts and ends. The initial tier could be the DNS servers used to resolve the site name, the firewalls protecting the site from direct access, hardware load balancers positioned in front of

the Web tier, or the Web tier itself. Also, some IP-based services such as Lightweight Directory Access Protocol (LDAP) servers are best represented as a single tier.

Key parts of this Service Model are its intra-tier and inter-tier aspects. Tiers help IT organizations both measure the service levels of e-business applications and respond to infrastructure problems. For example, one method for measuring service availability is to consider the service up if at least one element of each tier is active. However, the operator can define alternate availability measures as well. Similarly, tiers help IT organizations respond immediately to service level violations. By knowing how data is flowing through the service, a policy engine can more easily implement various policies to improve availability or performance. For example, an overload at the application tier might be avoided by throttling user access at the Web tier, thus giving the remaining users good performance. Also, an intrusion detected at the Web tier might be addressed by reconfiguring a firewall policy in a preceding firewall tier. These types of control policies would be difficult to implement without knowledge of the service's underlying tiers and elements.

### **Service Level Objectives (SLO)**

After creating a service, goals and priorities for each service should be defined. These are expressed as service level objectives (SLO). An SLO often includes the affected service, metric, threshold, conformance percentage, priority, and optionally exception periods and other restrictions. Here are two simplified examples:

Priority #1: The "Online Trading" service must have a user response time less than 5 seconds, 80% of time.

Priority #2: The "Online Support" service must be available over 99.99% of the time except on Saturdays and Sundays.

An actual SLO would need to have additional details such as how and from where to provide the measurement, the period over which to measure SLO compliance, etc. The operator controls how much is encompassed in the SLO by defining both the metric and its source.

For example, measuring the response time of a dynamic page may include testing the round-trip time from the Web tier to the application and database tiers. Alternatively, a WebLogic (BEA application) load test may measure the application tier in isolation. However, there are tradeoffs for different measurements: a test run external to the site may more directly mimic end-user experience, but may result in capturing a number of element problems over which the operator has no control.

An SLO is typically associated with a statistic or metric that is collected periodically to measure compliance. An active service level management solution should include a number of statistics collectors, or give the administrator the flexibility to provide custom metrics. A common approach is to generate a synthetic transaction that mimics end-users. These transactions run at regular intervals with the premise being that its performance will adequately reflect end-user experience.

A less common approach is to provide instrumentation of applications to more directly measure application performance. Resonate's solution combines aspects of both approaches. Because of its ability to see the initial network packets from the

client, and the client's request to close the connection, Resonate's solution provides a measure of overall transaction response time that includes the Web, application, and database tiers. Whatever the approach, it is of course critical to have a way to measure whether the service level is being met or not. A threshold is associated with the metric to define the actual goal. A high priority objective may have a flexible goal, while a lower priority goal may be more stringent. In this way you can not only ensure that high priority objectives are met, but also define how excess resources should be allocated.

Some transactions take longer than others, and there are a variety of causes. The transaction itself could be more complex, or there could be a temporary bottleneck in the Internet. To compensate for this, a conformance percentage can be specified that indicates how often the goal should be met for the SLO to be considered to be in compliance. There are also times when systems must be brought down for maintenance, upgrades or backups. Therefore, it is critical to define when the SLO should be monitored, and when it should be ignored. Finally, each SLO has an associated priority. Priorities provide explicit guidance on the handling of shared resources when SLO violations occur.

As a final step, the administrator can provide policies for the actions to be taken when SLO are not being met. Sophisticated policies take into account priorities, element roles, and availability and performance tradeoffs. A sophisticated policy engine is used to monitor SLO compliance, and activate policies when appropriate (policy management is discussed in detail later in this paper).

### **Service Monitoring and Reporting**

The Service Model itself is a useful tool for service monitoring and reporting. Instead of requiring an operator to manually associate element alarms with the impacted services, the process can be managed automatically.

Because element alarms can be abstracted to higher-level service alarms, it becomes possible for the trouble tickets to be service-specific as well. This can be critically important as administrator roles may be defined around service or application responsibilities.

An active service level management solution also takes advantage of the service model to hide configuration complexity. For example, by defining services, the monitoring system now knows which monitors and tests are needed and can auto-configure them. In one sense this is merely a substitution of one configuration activity for another, but the key difference is that the service model maps more directly to an administrator's job responsibilities, and therefore, should be more intuitive to use.

In a typical event management system, it is often very difficult to represent the complex relationships associated with service monitoring. The rule languages are often complex to understand. With active service level management, the complexity is reduced since many correlations are automatically implied by the structure of the Service Model. For example, using the above Service Model, the severity of a Web Server 1 (W1) failure depends on the availability of Web Servers 2 (W2) and 3 (W3). If one of those servers is active, then the problem is less critical than if all the servers had failed. Also, a failure of W1, W2, and W3 is more critical than failures of

W1, W2, and Application Server 1 (A1) since the latter case doesn't imply a loss of service availability.

Effective service monitoring involves a combination of event-based alarms with real-time and historical statistics. Performance bottlenecks and element failures are reported as alarms from throughout the distributed infrastructure. Statistics are collected to baseline tier and service performance, as well as to measure SLO compliance. Monitoring SLO can result in automatic compliance reports. Since IT managers are often evaluated on how well they meet their SLO, their compliance reports could become their initial management dashboard view. When problems occur, administrators can drill-down into affected services, tiers, and elements.

---

## Policy Management

In addition to service-based monitoring and reporting, the Service Model can be used for policy management. An approach to control that is based on prioritized goals or SLO is much more intuitive than a series of rules within a rule-based management system.

As described earlier, one or more SLO can be associated with a service. Policies are applied when objectives are not being met. Policies are an ordered list of corrective actions that are taken to fix a service level violation. It is important to note that a policy is attempted only when an SLO is not being met. This gives the administrator some level of confidence that the policy engine is taking appropriate actions. The policy engine obtains immediate feedback on the success of corrective actions by closely monitoring the SLO. If the SLO has not been attained, the next configured policy is attempted.

The overall process is defined in figure 3. After being configured with a list of services and SLO, the policy engine waits for various events and statistics to be received from the environment. The service level metric is collected at regular intervals for each SLO so that SLO compliance can be computed. If an SLO violation occurs, the policy engine will generate an SLO alarm. Tier-based or service-based performance statistics are also collected so that baselines can be established. This information is used for diagnostic purposes when problems occur.

Events from the environment include application or system bottlenecks and failures. The policy engine will correlate these events with the service model to create service alarms when appropriate. For example, a server failure may result in a "service degraded" alarm for each affected service. The policy engine will also use the element alarms to update its internal state representing the element status for all elements associated with services. This information is used later when attempting to apply policies to correct service level violations.

An ordered list of policies may be specified for an SLO. The following are some sample policies that might be utilized to address service level violations:

- Adjust traffic volume to specific servers (server weight)

- Reduce low priority traffic
- Activate backup server (just-in-time provisioning)
- Activate 'sorry' server
- Adjust traffic volume to specific sites (POP weight)
- Execute custom scripts

Each policy may be quite complex. The policy engine will compare its knowledge of the policy with its internal Service Model to see if a particular policy is applicable. Applicability includes both whether it might help meet this SLO as well as its likely impact on higher priority services that share resources.

As an example, adjusting server weight involves first evaluating whether there is a tier associated with a degraded service. This would be true if performance alarms had been received recently from at least one element in the tier. It might be particularly useful if only one element was having performance problems, but the other elements within the tier were not. A shift in traffic load balancing may balance the workload and eliminate bottlenecks. However, before making an adjustment the policy engine will also make a determination as to the likely impact any policy has on higher priority services. If a detrimental affect is predicted, then the policy will be skipped and the next policy will be considered.

This service-based approach to actively control service levels is more intuitive than conventional rule-based approaches. This new approach scales as new services are added because the relationship between services is driven by prioritized business goals instead of complex relationships between disparate elements.

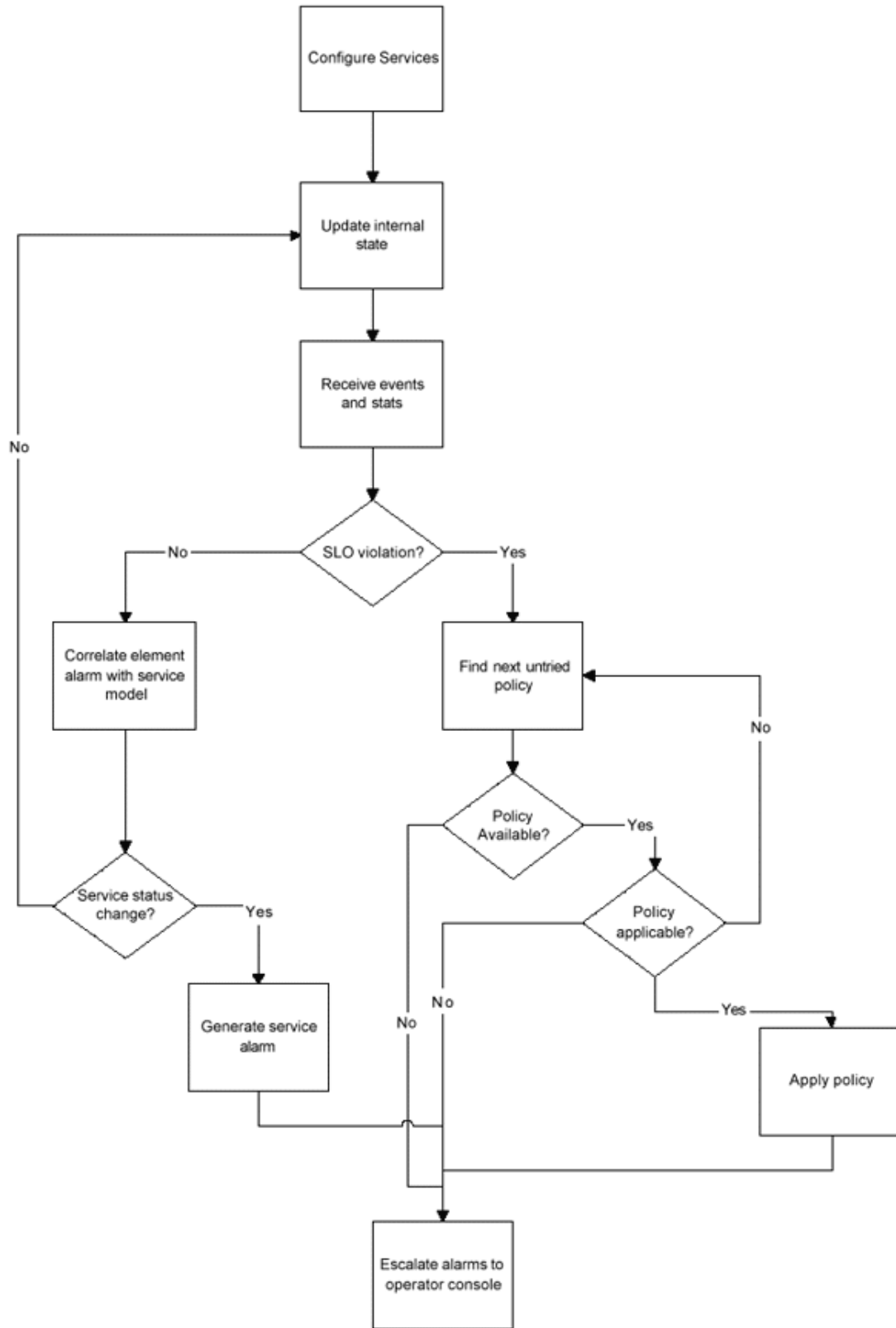


Figure 3. Policy Management Process

## **Policy Management Example**

Imagine a service provider delivering a "Stock Trading" service to a financial institution and a "Shopping Cart" application for an e-commerce company. Both services have stringent availability objectives and response time goals. The availability goals are most important.

The services are isolated from each other on separate systems, and follow the 3-tier model described earlier. There are also spare Web servers that could be used by either service. The Web servers have not been provisioned in advance for a couple of reasons. The service provider does not know which service will need the extra capacity, and also does not want to give their customer better performance without the customer paying for it.

An active service level management solution is used to define each of the services along with its respective availability and performance SLO. The spare servers are configured as backups in the Web tier of each service definition. A policy to activate a backup is associated with each performance objective.

The services are then monitored automatically. As long as the SLO are being met, no action is taken. However, if the performance of the Stock Trading service started to degrade due to a sudden spike in user demand, the SLO violation would cause the policy engine to look for an applicable policy to execute.

In this example, the Stock Trading service's Web tier is overloaded by the new user demand. The policy engine detects this with its system performance monitors deployed on the Web servers. The "activate backup" policy is executed whenever the Web tier becomes overloaded and a backup is ready. The "activate backup" policy is also sophisticated enough to not execute if it negatively impacts the performance of higher priority services.

A similar scenario might involve an overload at the application tier. A sudden increase in trading volume is handled by the Web servers, but the application servers are overloaded. In this case the "activate backup" policy would not apply, since no application server backups are defined. The policy engine will look for another applicable policy.

If no applicable policies are found, then the policy engine would simply report the diagnosed problem (application tier overload for the Stock Trading service) to the management console. However, because of the Service Model, sophisticated policies are possible. For example, an "activate sorry server" policy may be used. When Web sites are overloaded, it is becoming common practice to activate a sorry server that offers the user a coupon if he or she returns at another time. The pre-condition for this policy would be that the application tier is overloaded, and a sorry server is configured for a preceding Web tier.

---

## **Distributed Management**

As described above, e-business services and applications are distributed. To be most effective, critical management capabilities such as monitoring, diagnostics, and

actions, should be distributed as well. This allows the management solution to scale as the infrastructure grows, provide key information needed for fast problem resolution, and avoid overloading the centralized policy engine and console (Controller).

An important component of active service level management solutions are distributed agents that monitor each individual server so that system performance bottlenecks can be detected locally. Additional application-specific monitors deployed locally would collect and analyze application performance and availability metrics. These application-specific monitors also have built-in, application-specific correlations so that only important events are communicated to higher levels of the management solution architecture including the Controller.

These local application-specific and system monitors feed events and statistics to another type of local agent that handles network communication with the Controller. These agents are also responsible for providing "heartbeat" signals to the Controller ensuring a high availability environment for the management solution. Furthermore, these agents are able to run local tests and actions so that problems are isolated and fixed anywhere within the e-business environment. Finally, these local agents provide basic levels of event correlation and filtering across different systems and applications. The Controller provides additional levels of correlation for critical events that it collects from these agents. For resources where a local agent is not appropriate, the resource's behavior is monitored remotely through a proxy.

In addition to local monitoring and correlation, testing functionality is also distributed. Tests are executed at optimal locations to either a) mimic customer experience, or b) provide the necessary triangulation needed for the proper diagnosis of problems. The policy engine is located centrally within the Controller. Generally, controls should be centralized to handle the global prioritization of shared resources. It would be difficult to manage multiple competing policy engines.

Despite the desire for centralized policy management, there are additional elements of control that are distributed. For example, in addition to reporting events and statistics, the local application-specific monitors are also capable of restarting processes. Finally, each e-business tier is likely to have application-specific availability and control mechanisms. For example, Web servers are often controlled by software-based or hardware-based load balancers. In addition to balancing traffic, they can route around resource failures or bottlenecks.

---

## **Conclusion**

This paper presents active service level management as a complete and innovative approach to proactively manage the performance and availability of e-business applications. Enterprise IT organizations should consider implementing this type of management solution for their mission-critical e-business services if they are to assure end-user service levels while minimizing operational costs.

---

Brad Stone has vast experience in software architecture and project management. Prior to joining Resonate, he served as technical architect at Hewlett-Packard, which he was responsible for architecting workload, fault, and multisystem management solutions for HP-UX systems. He also served as consultant to lab and section management on system management issues including creating the visions for R&D roadmaps. Brad has published *Unix Fault Management*, a book describing how to monitor HP-UX and Solaris server components, including the applications, databases, and networks. Also at Hewlett-Packard, Brad served as R&D Project manager where he led a project team delivering high availability products for HP-UX cluster environments.

Brad holds a B.S. Honors in Computer Science from the University of Michigan and an M.S. in Computer Science from Stanford University.

Resonate is a registered trademark, the Resonate logo, Keeping E-Business Open for Business, Resonate Commander Solutions are trademarks of Resonate, Inc. All other trademarks are the property of their respective owners.

Copyright ©2001 Resonate, Inc. 09/01 IWP010