Intelligent Alarming and Monitoring

Whitepaper

Table of Contents

Preventing Catastrophe: Intelligent Alarming as a Lifeline for Contact Center Software

Application Uptime is more than a buzz phrase; it can represent the vital difference between success and failure, between a satisfied and unsatisfied customer, and between compliance and non-compliance. In a complex, interconnected system, downtime has a cascading effect. One service failing in a single application can negatively impact systems throughout the ecosystem. In a call recording environment, a single obscure error could indicate systemic issues that lead to loss of calls, productivity, revenue, and more.

The growing importance of application uptime has led many businesses to employ intelligent alarming as a solution and a service.

Call monitoring infrastructure can represent a significant cost center for any business, even those whose business is call monitoring. Onsite equipment, maintenance, and staffing represent a substantial investment just to ensure application uptime. In environments where monitoring applications are tied into workforce management and payroll, such as in a Verint implementation with the latest release, resource requirements around monitoring are even higher. Unfortunately, even basic deployments of call monitoring software come with a myriad of failure points, some more obscure than others. This has led to an increasing need for intelligent alarming, both as a solution and a service.

Compounding Complexity

Consider application services. A single server running call recording software may have more than a dozen different services (depending on the configuration and installation) running at the same time. Each service contributes to the overall function of the call recording software in a different way. One service, such as the *Recorder Agent Service* may have a specific role that applies in specific circumstances (screen recording, in this case). The *Recorder Agent Service* is not the only service that relates to screen recording; the *Screen Capture Service* is also part of the web of services that makes screen recording work. If the *Recorder Agent Service* stops does it mean vital information has been lost? What if it doesn't exist on a system at all? The interaction between these services is not always linear; it doesn't equal a 1 or a 0 as much as it equals *maybe* a 1 or *possibly* a 0.

Hyper-alarming is when software requires high amounts of bandwidth to review frequent, non-prioritized, alarms. It may result in alarm fatigue, which in turn leads to slow or no responses when critical systems fail.

To add additional complexity, server hardware can have an impact on call monitoring. A server running out of disk space is the most obvious example; when room to store data runs out, calls can be irretrievably lost. While it is important to recognize the obvious problems, the subtle ones represent a greater threat. Frequent, reoccurring events taken in the right context may predict hardware failure. They may also be symptomatic of a software-level issue that is having a business impact without anyone knowing.

Understanding the relationship between software and hardware is just one aspect of the application uptime puzzle. Hidden failures are another. Services can appear started but fail (hang), adapters may seem to communicate while losing an entire stream, and call tagging information may be lost while calls continue recording, all without raising the flags administrators need to become aware of.

The Value of Monitoring Software

In best case scenarios, monitoring can provide predictive information that prevents problems from ever manifesting.

Monitoring software, at its most basic level, is designed to draw attention to specific events. Most out of the box monitoring software uses industry-standard thresholds to determine the criticality and impact of a given event. For example: it is common to monitor memory utilization. When memory availability decreases below a certain threshold, the software raises one flag. If it decreases below that threshold for an extended period, it may raise an additional, or different, flag. In many instances, the software can provide some historic information about the event or even suggest ways to resolve the problem. In best case scenarios, monitoring can provide predictive information that prevents problems from ever manifesting.

Depending on the design and implementation, monitoring software can function like an additional pair of hands. The ability to send instant notifications via email, SMS, or a phone call when an alarm occurs provides technicians with the confidence they need to focus their attention elsewhere. Simple issues, such as when a service stops, can be managed by the software itself, alleviating the need for another resource to become involved. This is particularly useful in businesses where uptime is required around the clock or if there are staffing constraints.

Simple issues, such as when a service stops, can be managed by the software itself, alleviating the need for another resource to become involved.

Unfortunately, monitoring software is not without its limitations. To reach greater market saturation, the software is, by necessity, built to be somewhat generic. This means that a given software package is unlikely to provide predictive, actionable intelligence about a specific environment without a degree of customization. The more specific the business environment, the greater the customization required to make the monitoring engine effective. Poorly executed, monitoring software can lead to additional work and frustration, not less.

To customize a monitoring package, engineers need two key things: first, they need an understanding of the business and its infrastructure; second, they need an in-depth understanding of the software environment to be monitored. Due to advances in technology, a certain degree of business logic is built into most modern monitoring engines. Common elements, such as the ability to monitor hardware configurations, are nearly universal. That leaves the greater challenge: understanding the software environment. As mentioned earlier, complex software interactions greatly increase the difficulty of implementing a static monitoring solution. This is where intelligent alarming can add substantial value to a business.

Intelligent Alarming as a Solution and as a Service

Robust monitoring isn't about 1s and 0s; it's about thresholds and values that change depending on variables. Generic implementation works on generic systems. The call recording world is nowhere near generic.

Intelligent alarming refers to the process of customizing a monitoring engine as it applies to a specific suite of software. In a Verint environment, this means tuning the engine to include the variables that contribute to making the software work. Specifically, this tuning includes understanding the business environment and adjusting alarms to fit those needs. A discovery must be performed, so the engineers implementing the solution can do so accurately and quickly. Once the discovery is complete, monitoring can be implemented in a way that specifically and categorically relates to the business need. Finally, an alarming report must be provided to the business so they can understand what to expect and how to respond to alarms when they happen.

Call recording environments are never static. Simply implementing a well-configured monitoring solution does not guarantee it will continue to function optimally, particularly as the business landscape evolves. This is where intelligent alarming as a service can add substantial value. Once a monitoring solution is implemented, the service steps in to provide filtered alarms, post-implementation calibration, and ongoing development.

Alarm Filtering

Alarm filtering amounts to having additional staff dedicated to ensuring application uptime. Dozens, if not hundreds, of alarms can trigger daily in a base Verint implementation. Adding additional modules and functions only increases the noise. Intelligent alarming reduces the volume down to a more manageable level, but it isn't fool proof. Monitoring engines are only as capable as the logic they are programmed with. In businesses with multiple production servers in play, alarm filtering is an invaluable component of intelligent alarming as a service. Alarm filtering puts a subject matter expert (SME) between the initial alarms and the technician who will act on that alarm. Being an expert on the Verint software, the SME can identify if the alarm is actionable, categorize the priority (and therefore identify which notification channels are appropriate), and provide insight into the meaning behind the alarm. Combined with historic analysis and reporting, alarm filtering can provide predictive insight into business challenges and prevent downtime.

Post-Implementation Calibration

Post-implementation calibration is the process by which engineers review alarms for validity and adjust the monitoring engine to better suit business needs. Out-of-the-box monitoring of a Speech Transcription server is set to alarm when transcription falls below a certain number, or when transcription errors reach a specified threshold. After implementation, when changes to the customer environment take place, such as adding additional call centers, these numbers must be reassessed. In the case of a call recording environment, understanding the interactions of software, hardware, and thresholds can be incredibly complex; throttling a memory threshold too high may, for example, allow recording modules to go outside of operational ranges. Intelligent alarming services look for these types of events and provide businesses with the information they need to avoid becoming non-compliant or losing valuable data.

Ongoing Development

One step beyond ongoing calibration is development. A service seeks to continually improve and grow. Intelligent alarming as a service includes full-time development resources. These resources are devoted to customizing the monitoring engine, releasing continuous updates, and improving calibration tools. As best practices are defined, developers apply them across the solution, uplifting and enhancing stability to all constituents of the service. This is of incredible value in the Verint environment, as patches and version updates can dramatically change the way the system works. Large updates, such as version changes, propagate outward and impact both hardware and software. With dedicated resources who understand the impact of these updates, the opportunities for system failures can be significantly reduced. Preventative measures ultimately save businesses more than troubleshooting after the fact, particularly if litigation is involved.

Summary

Monitoring is a key element to any business infrastructure, however, it must be implemented appropriately. In most cases, out-of-the-box monitoring software can cover a variety of universal business needs without much customization. As business demands for maximum uptime and system availability grows and environments become increasingly complex, with hardware and software interacting in specialized ways, standard monitoring solutions simply do not suffice. Highly customized, business critical software, such as Verint Workforce Optimization application suite, require equally customized monitoring solutions. Intelligent alarming is the best, most cost-effective way to meet this need. Implementing intelligent alarming as a solution and service provides highly customized, software-specific solutions for the Verint environment. The service provides alarm filtering, post-implementation calibration, and ongoing development; each of which functions like adding multiple additional Verint subject matter experts to prevent future problems. Failure to implement proper alarming can result in loss of data and, at the worst, substantial fines.