solarwinds

IT'S AUTOMATION, NOT ART

by Leon Adato

TABLE OF CONTENTS

INTRODUCTION

We recently reached out to IT professionals to find out what they thought about monitoring and managing their environment. From the **survey,** we learned that automation was at the top of everyone's wish list.

SolarWinds has always worked hard to make monitoring as elegantly simple as possible, which means making a lot of things work right out of the box (ie: automatic discovery and monitoring of devices, applications, and more). So we know how big a subject this is. But we also understand that the elegance of a feature can often be masked by its simplicity, in fact, the simplicity is PART of the elegance, and all of it, when done well, can appear to be so effortless that it seems trivial.

Movie history buffs may recall that Fred Astaire's first audition for MGM included the appraisal, "Can't act. Slightly bald. Can dance a little." Later comments from directors and co-stars explained that Mr. Astaire's grace and committment to perfecting each movement created an air of effortlessness that could fool the casual observer into believing that his dancing was simplistic.

Here at SolarWinds, we know how he must have felt.

ABOUT THIS GUIDE

This guide was written to provide an overview on automation as it relates to monitoring. It was designed specifically for those familiar with computers and IT, who know what monitoring is capable of, and who may or may not have hands-on experience with monitoring software.

If you find yourself a bit behind on monitoring concepts, we can recommend the free and thoroughly comprehensive Monitoring 101 Guide, found **here.**

This completely tool-agnostic guide will not ask you to actually dive into monitoring automation using software package XYZ, but it is my hope that after reading this, when you do start setting up automation using monitoring software XYZ, some of the functions will be less arcane.

ABOUT THE AUTHOR

"I have only made this letter longer because I didn't have time to make it shorter."
– Blaise Pascal

In a career spanning three decades and four countries, Leon Adato has been an actor, electrician, carpenter, stage combat instructor, pest control technician, Sunday school teacher, and ASL interpreter. He also occasionally worked on computers.

Leon got his start providing computer classes, worked his way up the IT food chain from desktop support to server support to desktop environment standardization engineer and onward, to systems monitoring, management, and automation. Along the way he discovered a weird love for taking tests, and picked up an alphabet's worth of certifications, including WPCR, CNE, MCP, MCSE, MCSE+I, CCNA, and SCP.

Focusing on monitoring, Leon spent almost 20 years honing his skills at companies that ranged from big (National City Bank) to bigger (CardinalHealth) to ludicrous (Nestle), becoming proficient with a variety of tools and solutions along the way.

A user of SolarWinds software since 2003, Leon attracted the attention of SolarWinds staff via his impressive participation in **THWACK** forums, including providing helpful posts, attending UX sessions, taking part in beta and RC testing, and whining.

It was about that time that Head Geek Patrick Hubbard noticed that Leon was long-winded to a fault, and that he lacked any semblance of self-restraint or basic common sense when it came to speaking in front of large audiences. That led to his being offered a position as Head Geek in 2014. Leon has never looked back since, mostly because he's incredibly uncoordinated and would surely run into something.

AND NOW, A WORD FROM OUR SPONSOR

I've been working in IT for over 30 years, since 1989. For nearly 20 of those years I have focused on monitoring, management, and automation in the enterprise. And I've used SolarWinds tools since 2003 (although not continuously). So here's what I've learned in all that time:

First, SolarWinds solutions are geek-built. What I mean by that is that the solutions we provide are made by geeks (otherwise known as sysadmins, engineers, IT professionals, and even Head Geeks), for geeks, to solve real problems in the workplace right now. We spend copious amounts of time talking to people in the trenches to find out what problems they are having and how they would like to see them addressed. We take that feedback and turn it into the list of features we build into the next version.

Second, SolarWinds solutions are modular. You don't need to get a whole suite in one monolithic installation. You can determine which functionality you need and apply only the modules that meet those needs. The modules, which integrate well with solutions from other vendors, snap together under a common framework. Real geeks know that you don't get to pick every single piece of software the company uses, and that, like a good mixed-breed dog (ie: a mutt), heterogenous solutions are often the most robust and faithful allies you can have in the data center. The flipside of this is that each module is flexible. Each tool has a variety of outside-the-box actions you can take to get almost any job done.

Finally, and there's no way to dress this up, SolarWinds solutions are chea... uh... affordably-priced. Especially when you consider the features in each module. Head over to **www.solarwinds.com** for more detailed information on products and pricing; or to download a free, unlmited (meaning you can load up as many devices as you want), 30-day demo of any (or all) of the SolarWinds modules.

PRO TIP: there's also about two dozen free tools you can download over at **www.solarwinds.com/free-tools**

DANCE OF THE SPECIAL SNOWFLAKES

In the world of IT, it seems to be a commonly held belief that your environment is somehow special – measurably and materially different from all those other companies. "You don't understand," you find yourself explaining to vendors and other IT pros. "We're different." And then you go on to list attributes of your company that actually apply to 99% of the businesses out there. With this type of attitude, the speaker justifies how best practices, common techniques and standard solutions don't apply, or at least must be significantly altered to fit the delicate and beautiful snowflake that is their IT architecture. And nowhere I have seen this perception as vehemently declared as when the conversation rolls around to monitoring.

I've lost count of the number of organizations who insist they require an in-house, custom-crafted, artisanal solution that more resembles an interpretive dance than a technology, and, of course, requires special care and feeding by either a lone hermit/wizard/engineer who only speaks in enigmatic koan; or a mystical cabal of specially trained SysAdmins who were raised by Linux-wielding monks in a far-flung technical monastery.

Many monitoring solution providers don't help matters, each spinning tall tales about the way they leverage special APIs and context-sensitive command sets, that somehow only they know, due undoubtedly to the combination of their great skill, long beards, and certifications from Hogwarts-like institutions.

With 30 years in IT and almost 20 of those focused on the monitoring space – not to mention having used just about every solution on the market since 1998 – I'm here to tell you a little secret I've learned:

Monitoring is simple.

To be sure, implementing good monitoring can be hard work. GOOD monitoring, which is robust enough to collect the statistics you need without injecting observer bias, is hard work. But, essentially, it is simple in its fundamental essence.

And of all those simple-but-not-easy aspects of monitoring I just listed, one seems to be more elusive than all the others: automation.

MONITORING MADLIBS, PART ONE: MONITORING IS A ______

First, let's get something out of the way: monitoring is not a ticket, or a page, or a screen. Monitoring is nothing more than the ongoing, consistent collection of metrics from and about a set of devices. Everything else – reports, alerts, tickets, and automation – are a happy by-product of the first part.

So Job One is to make sure you are collecting all metrics, including interrupt-based messages, like SNMP-trap and syslog; ad hoc polling, like WMI and SNMP-get; protocol-centric options, like IPSLA and NetFlow; simple techniques, like ICMP (ping); complex solutions, like Deep Packet Inspection; and even vendor-specific techniques that take advantage of a system's API.

Job Two is setting alerts for when things go off the rails. Doing so lets you know when a system is down, it gives you a record of when a router configuration was changed, and much more.

As a monitoring engineer, my second-favorite thing is listening to colleagues answer the question, "How do you know something is wrong?" It makes them define "bad" for themselves, which leads to good monitoring and alerting. My most favorite thing, though, is following that with, "So, then what? When it's bad, like you just described, what do you do about it?"

The answer to that questions sits at the heart of the conversational path that leads to automation.

THE SIREN SONG OF THE BLACK SWAN

In business, when an unforeseen and unexpected failure occurs, management often acquires a dark obsession with post-analysis. Postmortems are called with the express intent of ensuring that that particular failure never happens again. Time is spent on figuring out what went wrong, but even more time is spent on finding a way the event could have been predicted.

But if the event was a so-called Black Swan, a term coined by economist Nicolas Taleb to describe things that cannot, by their very nature, be predicted either because they are simply too extreme or because no current source of data has the applicable insight – the post-analysis exercise is thoroughly unnecessary.

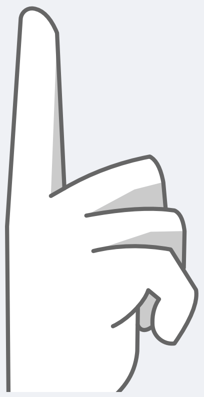
I'm not saying that businesses shouldn't attempt to learn lessons from their failures. But they should try to determine whether the failure was predictable in the first case. If it wasn't, there is no point in conducting a postmortem. **Hunting down black swan events is often a waste of time and money.** A single big and unpredictable event distracts everyone from the common, everyday, and ultimately more expensive and solvable opportunities around them

WHY AUTOMATION?

With a perfectly good monitoring system, you may wonder why automation is so important.

Because you are lazy. Because most seasoned IT professionals are lazy. Because spending your day responding to tickets, alerts, and emails is both tedious and boring.

Because your time is valuable. Too valuable for this.



Remember my favorite question? The one I said I loved to ask whenever someone told me how they know when something is bad? "Okay, then what?"

Maybe after they get an alert, they clear a queue. Or restart a service, or delete all of the files out of a temp directory.

Whatever it is, it's very likely going to be something that could have been run without human intervention if the action isn't already built into the monitoring tool itself. That's a quick win for automation, as long as it always solves the problem. As anyone who has been working in IT for more than 15 minutes should know, while many problems are tediously repetitive, sometimes you get the one weird case that resists all your usual tricks.

This is why sophisticated monitoring solutions will allow you to build an alert that triggers an initial action, then waits a specified amount of time. If the condition persists, a second (or third, or fourth, or whatever) action will be triggered.

Thus, a disk full alert could first clear the temp directory, wait 10 minutes, then remove specific application log files more than one month old, wait another 10 minutes, create a ticket for the server team, wait one hour, and, if the problem continues to persist, page the team lead as an escalation point.

But let's say that there isn't a definitive action that can be taken. Maybe the answer is to check the last 15 lines of this log file, look at this other counter, and run a test query from the application server to the database. Based on the results of that information, the technician will know what to do next.

In that case, your automated action is to do all those steps and then insert those results into the alert message.

Then, instead of a message that says "Service XYZ is down," the technician receives a ticket that already contains greater insight into the conditions at the time of the failure, as opposed to 15 minutes later, when they've dragged their butt out of bed, fired up the laptop, and started to dig into the situation. By doing so, the monitoring system has effectively given staff 20 minutes of their life back. It has simplified the troubleshooting process.

And the best part of all this is that even if the information you pull isn't 100% necessary, the reality is that the information is useful most of the time. Gathering that information proves that the monitoring system can be leveraged as an always-on, Level One diagnostician.

Trust me when I tell you that doing this kind of thing will make heroes out of you and the monitoring system.

MONITORING MADLIBS, PART TWO: AUTOMATION IS FOR _____

Now that we know what monitoring is and why automation is a good idea, we should look at some of the areas where automation brings solid, measurable value to the monitoring discipline within IT.

As previous sections have alluded, automatically responding to issues is a big win, but there are less obvious, yet equally important aspects of automation. Let's take a look at them now.

Discovery, Part One

One of the first things you do when you fire up a new monitoring system is load devices. While this can be done manually by specifying the name or IP address of the machine and connection information, in any environment larger than about 20 systems, this becomes an exercise in tedium and frustration. Scanning the environment is far more convenient.

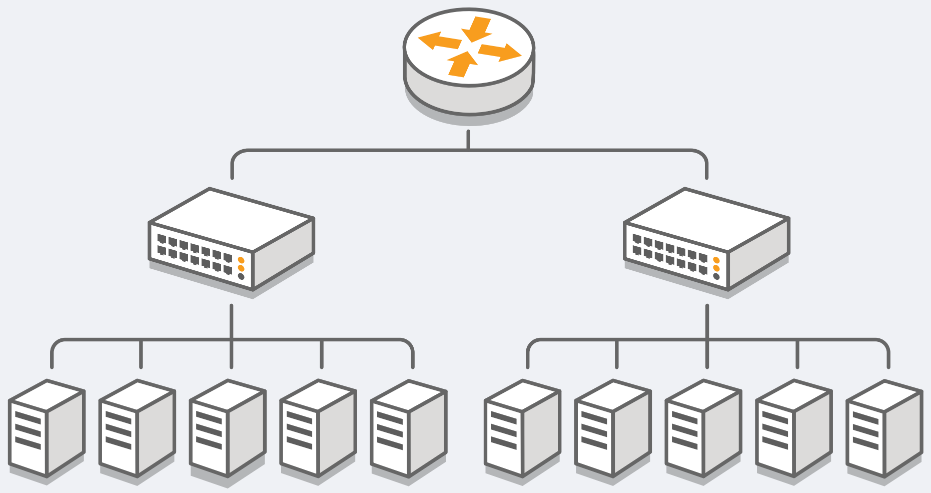
For about a week.

After that, again, in environments larger than 20 systems, devices have most likely been added, changed, or removed. And manually kicking off a scan of the environment, which includes reviewing the results, weeding out the redundant systems, selecting interfaces, etc., gets old after the third week.

This is often where the itch for automation starts.

Connectivity

Discovering a bunch of devices is good. Knowing how they all connect is better. Why? Let's imagine a simple scenario where we have one router connected to two switches, which are connected to 10 servers.



If the router goes down, how many **device down alerts** should you get? That's right: one. For the router.

Now, how many alerts will you get?

If you answered 10 (for the servers), 12 (for the servers plus the switches), or 13 (for the whole shootin' match), you may be right and at the same time so very, very wrong.

The solution is to set up monitoring so that when the router goes down, the rest of the downstream devices are put in an unreachable (ie: NOT-down) state.

While this can be done manually, it would be so much better to have a monitoring tool that goes out and scans the environment and all the rest for you.

Discovery, Part Two

You've found every device that was hiding in the nooks and crannies of your network. You've scanned them to find out how the router-bone is connected to the switch-bone. You've also established your virtualization hierarchy, from VM to host to cluster to data center. You've even determined what kind of hardware each device is, how many interfaces and drives they have, and the status of their physical hardware, such as fans, temperature sensors, RAID controllers, and more. You've got it all figured out.

Now if you could only remember which of those little stinkers is running your Exchange email service.

That's right. This discovery is all about software. As with Discovery, Part One, how this is done is less of a point of focus in this guide than the how often. Because, sure, you can scan a server when you add it and determine that it IS running Exchange or SharePoint® or your company's custom app, but you'll never know (until it's too late) that three weeks later the app developer enabled IIS. Or worse, DHCP.

If anything, you need automation to scan for applications even more than you need it to scan for new, changed, or deleted devices on your network.

But it's actually more complicated than that.

Monitor Assignments

It's one thing to know that a server is running IIS (or DHCP, or SharePoint, etc). It's another thing to understand what your company thinks about that server. Is it a critical SharePoint box, or just a QA server? Is it currently a new build that is being tested, or is it full production?

In many organizations, a server will go through a build and staging process before moving to production. At the other end of the cycle, a server may spend significant time in a decommission state where it is running, but not in use except in emergencies. Similarly, some companies have systems that could migrate from test to QA to product and/or back again.

In each of those cases, the intensity of the monitoring may change drastically.

So the automation opportunity here is not so much detection as it is applying the correct templates based on custom variables. You want to empower the owner teams to set these attributes, and then have the monitoring system detect those settings and automatically adjust the monitoring accordingly.

THE ELEPHANT IN THE ROOM

"Strategy is finding broccoli on sale and buying it for dinner. Tactics are getting your kids to eat it."

– SolarWinds® Head Geek™ Thomas LaRock

Before we dig into automation, and specifically about automatically responding to alerts triggered from your monitoring solution, there is a point that absolutely must be made. In some circles, it's called the detection-prevention-analysis-response cycle. What we've been talking about so far is the detection, (or monitoring), and response, (or automated remediation), aspects. But it would be negligent to ignore prevention and analysis.

Alerts are our way of catching errors when they occur. But then it's up to us to determine why they occurred and find a way to keep them from occurring again. When an alert is created, we have to do the hard work of analyzing the situation to find a pattern and root cause. This leads us to taking actions, such as creating views and reports, to help ensure the issue is prevented in the future.

Automatic responses to alerts keep your business running and help ensure you get the beauty sleep you need, but in the morning you and your team need to be able to see that something happened and do the hard work of figuring out why it happened, so you can prevent it from happening in the future.

I'M SCARED

"I must not fear. Fear is the mind-killer…"
- Litany Against Fear from Dune, by Frank Herbert

Standing at the edge of the ocean that is automation can be a little daunting. But you have to trust me when I tell you that the water is fine, you won't drown, and you have the ability to try things one step at a time. This is not an all-or-nothing proposition where the risk is that you'd go from zero to all-my-systems-are-offline.

Next, many monitoring solutions can do what I'm describing. If you find that yours can't, then you should consider why you are using it. Maybe your business has a really solid justification for remaining on that platform. At the same time, the capabilities I'm discussing here are standard.

As with any work in IT, having a plan of attack is sometimes more important than the attack (or in this case, the automation) itself. So let's run down a few things:

- **Identify your test machines first** – Whether that is lab gear set aside for the purpose or a few less-critical volunteers, set up your alert so that it only triggers for those machines.
- **Learn to use reverse thresholds** – While your ultimate alert will check for CPU>90%, you probably want to avoid spiking the test machines repeatedly. So just turn that bracket around. CPU<90% is going to trigger a whole lot more reliably, at least we hope so.
- **Find the reset option** – Closely related to the previous point, know how your monitoring tool resets an alert so it triggers again. You will likely be using that function a lot.
- **Verbose is your friend** – Not at cocktail parties or sitting next to you at the movies, but in this case, you want to have every possible means of understanding what is happening and when. If your tool supports its own logging, turn it on. Insert, "I'm starting the XYZ step now" messages generously in your automation. It's tedious, but you'll be glad you did.
- **Eat your own dog food** – If you were thinking you'd test by sending those alerts to the server team, think again. In fact, you aren't going to send it to any team. You're going to be getting those alerts yourself.

- **Serve the dog food in a very simple bowl** – You really don't need to fire those alerts through email. All that does is create additional delays and pressure on your infrastructure, as well as run this risk of creating other problems if your alert does kick off 732 messages at the same time. Send the messages to a local log file, to the display, etc.
- **Share the dog food** – And then you can share them with the team as part of a conversation. Yes, a conversation.
- **Embrace the conversation** – This process is going to involve talking to other people. Setting up automation is collaborative, because you and the folks who will live with the results day in and day out should agree on everything from base functionality to message formatting.
- **Set phasers to full** – Once the automation is working on your test systems, plan on a phased approach. Using the same mechanism you did to limit the alert to just a couple of machines, you are going to widen the net a bit, maybe 10-20 systems. And you test again to observe the results in the wild. Then you expand out to 50 or so. Make sure both you and the recipients are comfortable with what they're seeing. Remember, by this point the team is receiving the regular alerts, but you should still be seeing the verbose messages I mentioned earlier. You should be reviewing with the team to make sure what you think is happening is what is really happening.

Following those guidelines, any automated response should have a high degree of success, or at least you'll catch bad automation before it does too much damage.

Beyond the suggestions in the next section, a good rule of thumb for automating is to find the things that have the biggest bang for the least effort. A great place to start is to look at your current help desk tickets. Whatever system-based events you are seeing the most now, that's probably where your biggest impact can be had.

Another good place to to find ideas for automation is the lunch room. Listen to teams complain and see if any of those complaints are driven by system failures. If so, it may be an opportunity for automation to save the day.

Finally, don't plan too far ahead. As apprehensive as you may feel right now, after one or two solid (even if small) successes, you will find that teams are seeking you out with suggestions about ways you can help.

A CHILD'S GARDEN OF AUTOMATION

By now, you should have a solid understanding of monitoring and alerting. Now you can appreciate the value it has potential to bring to your systems, teams, business, and career. You have plans in place to identify potential automation opportunities, and the methodology for developing, testing, and rolling them out.

Next, I'd like to share some examples that I and the participants on THWACK have come up with over the years. Don't look here for specific trigger examples or code snippets. I'm simply providing a sketch from which you can let your imagination run wild.

Disk Full

At this point, you've heard about this enough that you can probably write it yourself already, but here are some specifics:

First, get the alert right.

As fellow Head Geek Thomas LaRock and I discuss in this episode of SolarWinds Lab, simplistic disk alerts help nobody. If you have a 2 TB disk, alerting when it is 90% used translates to having 204.8 Gb of disk spaces remaining. A good solution is to check for both percent used and also remaining space (as we do in the video). A better solution is to include logic in the alert that tests for the total space of the drive, so that drives with <1Tb of space have one set of criteria, and drives with >Tb have another (all within the same alert, if possible, because who wants to manage hundreds of alert rules?)

Regardless, you want to ensure that you are monitoring disk space in a way that is reasonable for the volumes in question, and that only create necessary alerts.

Next, clear unnecessary disk files out of various directories. For the purpose of this document, we're just going to say that all systems have a temporary directory (**`/tmp`** on Linux®; **`C:\windows\temp`** and **`C:\Users\<username>\AppData\Local\Temp`** on Windows®) and that we can delete all files out of that folder with impunity.

The challenge in doing this easily comes down to a problem of impersonation. Many monitoring solutions run on the server as the system account. This means that performing certain actions require the script to impersonate a privileged user account.

There are a variety of ways to do this, which is why I'll leave the problem here for the reader to solve in a way that best fits their individual situation.

Once the impersonation issue is resolved, there's another challenge specific to the disk full alert, and that is one of knowing that the correct directories for this specific server are being targeted.

The best way to do this is to utilize a common shared folder that is mapped to all servers, and place a script file there. That script can be set up to first detect the proper directories and then clear them out with all the necessary safeguards and checks in place to avoid accidental damage.

Restart an IIS Application Pool

Sadly, restarting application pools is often the easiest and best fix for website-related issues. I'm not saying that running "appcmd stop…" and then "appcmd start… " from the server command line is a quick kludge that ignores the bigger issues. I'm saying that often, resetting the app pool is the fix.

If the web team finds itself in this situation, then waking a human being to do the honors is absolutely your most expensive option. However, restarting the application pool is made slightly more challenging by the fact that one server could be running multiple websites, which in turn have multiple application pools. Or, you could have one big application pool controlling multiple websites. It all depends on how the server and websites were configured and you have no way of knowing.

If your monitoring solution can monitor the application pool itself, it will provide the name for you. Most mature monitoring solutions do this already.

Once you have the name, you can:

- Use the built-in **`restart application`** pool option that is now included in most sophisticated monitoring tools
- Run this command from the command line of the affected server:

  `appcmd [stop/start] apppool /apppool.name: <app pool name goes here>` **`appcmd.exe`** may not be in the path. You can typically find it in **`C:\windows\system32\inetd\appcmd.exe`**

- Appcmd.exe can't run against a remote system, so you will have to use another utility, like psexec to get this to run from the monitoring server against a remote machine.
- You can run a PowerShell® script locally or remotely, and the code would look like this:

```
# Load IIS module:
Import-Module WebAdministration
# Set a name of the site we want to recycle the pool for:
$site = "Default Web Site"
# Get pool name by the site name:
$pool = (Get-Item "IIS:\Sites\$site"| Select-Object
applicationPool).applicationPool
# Recycle the application pool:
Restart-WebAppPool $pool
```

Restart IIS

Running a close second is resetting IIS itself. This is, of course, the nuclear option in terms of website fixes since you are bouncing all websites and all connections. Despite the fact that it's drastic, it is a necessary step in some cases.

As with restarting application pools, getting a human involved to do this incredibly simple action is a waste of everyone's time and the company's money. It is far more advantageous to automatically restart and then re-check the website a minute or two later. If all is well, then the server logs can be investigated in the morning as part of a postmortem. If the website is still down, then it's time to send in the troops.

You can restart the IIS web server in a number of ways:

Use the **`restart IIS`** option in your monitoring solution (many of the better tools have this built in)

- Execute **`iisreset/restart`** at the local command line of the affected system
- Remotely execute **iisreset <computername> /restart**
- You can create an execute a PowerShell command:

  `invoke-command -scriptblock {iisreset},`

  or, more simply, use the call operator: :

  `& {iisreset}`

Restart a Service

Sometimes services stop. Sometimes, they are even services that you, as a monitoring specialist, care about, like SNMP.

So you are cutting dozens of service-down alerts. Have you thought about restarting them? In some cases, a restart doesn't really help matters much. But in far more situations it does. Computers are funny things.

"Screws fall out all the time. The world is an imperfect place."

– The Breakfast Club

Sometimes they just need a gentle nudge. If this is the case, you can:

- Use the **restart service** action that is built in to many robust monitoring solutions
- Issue the command **`net start <servicename>`** on the local computer
- Issue the command **`sc <computer> start <servicename>`** from a remote machine

- Run a PowerShell script with the following commands:

  ```
  (get-service -ComputerName <computername> -Name
  <servicename>).Start()
  ```

- For Linux systems, run the pkill command, either locally:

  ```
  pkill -9 <process name>
  ```

  or remotely

  ```
  ssh -l <username> <computername> 'pkill -9 <process name>'
  ```

To Kill a Process

It's also important to point out that there are other times when you might want to stop and then start a service or process. Such as when it is clearly running away (ie: RAM and/or CPU spikes to 100% (or climbs steadily over time), growing number of spawned instances, etc.)

While those conditions are slightly more involved to detect (but I have to say it again, most robust monitoring solutions today do this out of the box), the process to stop and start is identical to what you see above.

However, it's worth noting that, while crashed services or processes have a more immediate impact on the end-user experience (usually because the application no longer responds, or responds as expected), runaway processes are more insidious, causing the application to respond slowly or erratically.

And as we like to say at SolarWinds, "Slow is the new Down."

The other point to make is that restarting services or processes based on performance allows you to bring baselining to bear. With baseline options, the monitoring solution looks at past performance over time to define normal. Then, rather than setting a fixed threshold ("when RAM utilization is > 90%") you can set a threshold of "when RAM utilization is 40% higher than normal for this time period." This level of sophistication allows you to avoid triggering a false alarm when the processing spikes for entirely reasonable, normal, and predictable causes.

Restart a Server

If restarting the IIS service is the nuclear option, then restarting the entire server is akin to nuclear armageddon. And yet, we all know there are times when restarting the server is the best option, given a certain set of monitorable conditions.

Presuming your monitoring solution doesn't support a built-in option for this, some options include:

For Linux, issue the command:

```
ssh -l <username> <computername> 'shutdown -r now'
```

On Windows, you can remotely restart a machine by issuing the command:

```
shutdown /r /f /t 0 /m \\<machinename> /c <comment to add to
eventlog>
```

Or with PowerShell, you can do it with the
`restart-computer <computername> commandlet`

vMotion a Machine to a Different Host

From failing hardware to noisy neighbors, there are a variety of issues that could impact a virtual machine. The important point to understand is that the problem has nothing to do with the application or the virtual machine itself.

Luckily, many sophisticated monitoring solutions have the ability to detect these conditions. But what do you do about it when the issue has been detected? The standard response is to engage the server or virtualization team to analyze and remediate. But there's a simpler option that has no potential impact to the application, and in many cases can help. I'm talkin about vMotion®. This is where you migrate the virtual machine to a different host, away from the noisy neighbor or misbehaving hardware.

Sometimes problems follow the VM. However, if the condition persists, a ticket can still be created and the support team engaged. Plus, knowing that this has been tried (and tried automatically at any hour of the day or night) gives the responder that much more information to go on when they begin the work of fixing the issue.

While many monitoring solutions have a built-in option to perform a vMotion operation, you can still achieve it using the VMWare **`SVMotion`** command line option, or with the PowerShell **`MoveVM`** cmdlet.

Storage vMotion

Similarly, you can migrate the VM from one data store to another. While most of the situations that call for you doing this are operational in nature (moving to a new hardware vendor or from Generation N storage to Generation N+1), there are also conditions when you might want to SvMotion as a response to trigger an alert.

It's also important to note that there are several requirements and limitations when performing SvMotion, so you'll want to make sure there's never a chance to violate these before you set up automation that could destroy your VM. You can find more information about these requirements and limits **here**, also **here** and **here.**

To perform an automated SvMotion, you run the following command on the vSphere server:

```
svmotion
--datacenter=<datacenter_name>
--vm <VM config datastore path>:<new datastore>
```

For additional command line details and options, see **here.**

Back Up a Network Device Config

Everything I've gone over so far covers direct remediation-type actions. But in some cases, automation can be defensive and informational. Network device configurations are a good example, in that it doesn't fix anything, but instead gathers additional information to help the human fix the issue faster.

It's important to note that between 40–80 percent of all corporate network downtime is the result of un-authorized or uncontrolled changes to network devices. This isn't always due to malice. Often, it's simply a case of the change not being reviewed by another set of eyes, and an otherwise simple error getting past the team.

So, having the ability to spontaneously pull a device configuration based on an event trigger is super helpful.

To do this, you can either

- Copy the config with built-in functions in the monitoring solution, or
- Copy the config with PowerShell:

```
New-SshSession <device_IP> -Username <username>
-Password "<password>"
Results = Invoke-Sshcommand -InvokeOnAll
-Command "show run" | Out-File "<filepath and filename>"
Remove-SshSession -RemoveAll
```

There are two general cases when you would probably want to execute this automated action:

The first is when your monitoring solution receives a config change trap. While the details about SNMP traps are beyond the scope of this document (although you can check out the extremely comprehensive eBook, **Monitoring 101**, for loads of information on that and more), suffice to say that you can configure your network devices to send spontaneous alerts based on certain events. One of which is that the configuration has changed.

The second is when the behavior of the device changes drastically. Such as when ping success drops below 75%, or ping latency increases. In either case, often the device is in the process of becoming unavailable. But in some situations it is wobbly, and there's a chance to grab the config before it drops completely.

In both of those situations, having the latest configuration provides valuable forensic information that can help troubleshoot the issue. It also gives you a chance to restore the absolutely last-known-good config, if necessary. And if this leads you to think, "Well, if I have the last known good configuration, why can't I just push that one back?" then you, my friend, have caught the automation bug! Run with it.

Reset a User Session

Somewhere in the murkey past, the first computer went online and became node-1 in the vast network we now call the internet. The next thing that happened, mere seconds later, was that the first user forgot to log off their session and left it hanging.

For any system which supports remote connections–whether that is in the form of telnet/ssh, drive mappings, or RDP sessions–having the ability to monitor and manage remote connection user sessions can make running weekly, if not daily, restarts,unnecessary. Or at least that much smoother.

For Linux, use the **`who`** command to discover current sessions, or with greater granularity by remotely running

```
netstat -tnpa | grep 'ESTABLISHED.*sshd'
```

Once you have the process ID, you can kill it using the command described in the previous To Kill a Process section.

For Windows, you get the active sessions on a system using the **`query session <servername>`** command, and disconnect the session via the reset session **`<Session name or ID> <servername>`** command.

Or you can use the PowerShell cmdlet **`Invoke-RDUserLogoff.`**

Clear DNS Cache

There are times when a server and/or application will start misbehaving because it can't contact an external system. This is either because the DNS cache, the list of known systems and their IP addresses, is corrupt, or because the remote system has moved. In either case, a really easy fix is to clear the DNS cache and let the server attempt to contact the system at its new location.

In Windows, you use the command **`ipconfig /flushdns.`**

On Linux, the command varies from one distribution to another, so it's possible that **`sudo /etc/init.d/nscd`** restart will do the trick, or that you should run **`/etc/init.d/dns-clean;`** or that another command is needed. Research may be necessary for this one.

AN EXTRA FLOWER FOR YOUR GARDEN: SCRIPT WRAPPERS

One of the challenges with all these automation ideas is that the team who owns the system may want to add features, safeguards, or steps later. While the common approach is to make this part of the ongoing conversation between you and those teams, there is another option: script wrappers.

The concept of this is simple: When an alert is triggered, instead of issuing the commands as described in this section, you will instead execute a simple command that effectively says, "Go to the target system and run this other script."

For example, to clear the disk drives, you may simply execute the command:

```
\\SERVERABC\utilities$\monitoring\windows_disk_cleaner.bat C
```

What that's saying is to run the script **`windows_disk_cleaner.bat`** found on **`SERVERABC`** in the utilities share, **`\monitoring`** directory. One variable (C) tells the script to check the **`C:\`** drive. Once launched, this other script does the actual disk cleaning.

What does that achieve? It gives the owning group the power to change things down the road. You, the monitoring engineer, will always just launch that script. If it's not present on the server, nothing happens. If the code says to erase the **`\Windows`** directory, then that's what is going to happen. And if the team changes its mind later, someone just changes the script to get a different result without having to submit a new script to you for testing, pilot, rollout, etc.

It is a bit of a double-edged sword, but if you are working with an engaged group, who have a respectable level of coding experience and a vested interest in making automated responses as robust as possible, you may benefit from trying this out.

AUTOMATION ANSWERS

Now I'd like to dive into some aspects of monitoring automation in more detail.

Device Discovery and Connectivity

There are three aspects to this functionality that we need to discuss:

1. The ways new devices can be discovered.
2. The way a discovery can be executed automatically.
3. What to do with devices when they are found.

Every monitoring solution has its own spin on executing a device discovery. So let me begin by telling you about one set of techniques you should avoid:

First, the only way to discover devices is by using **ping** (ICMP packets) to sweep an entire network subnet (192.168.1.0/24, for example). There should be no facility for partial scans, or fixed lists of IPs. Not only that, but there's no way to indicate IP addresses to avoid. Second, each device that is found should be absolutely attacked by SNMP using the most aggressive method possible. We're talking a full SNMP walk of the entire device.

Why is all of that bad? Well, first because even that first time you run discovery on your network, you want to be thoughtful about it. If you have no intention of monitoring PCs or IP phones, why would you waste time scanning them? Scanning a network should not be an all-or-nothing choice.

Second, because that kind of aggressive behavior can often send the target system into a tailspin (or more accurately, a CPU-spin) and cause them to lock up, or worse.

WAYS TO DISCOVER

That covers what not to do. So now let's discuss the positive. You should look for solutions that let you scan your network in a variety of ways, combining the ability to:

- Enter a subnet
- Enter a list of IP addresses
- Enter a seed device where the network is discovered by finding connected devices from that first one
- Specify an Active Directory® OU, and scanning computers in that OU

The system should also, as mentioned earlier, provide the ability to specify a system that should NOT be scanned, and that system should take the exact same input as the discovery itself, either a subnet, list of IPs, etc.

The result is a robust discovery profile that only looks at the parts of the network you and your team want scanned.

Once a set of devices are found, the discovery should use the gentlest possible approach, interrogating a few basic pieces of information, such as the device name, vendor, and model. That information should then kick off a discovery of specific elements unique to that vendor/model, rather than a walk of every known numeric combination.

In addition, device scans should be able to utilize protocols other than SNMP, including WMI, vendor APIs (Cisco® UCS, VMWare®, and Microsoft® Hyper-V®, to name a few).

Finally, part of the scan should determine connectivity to other devices.This should reveal everything from which switch a server is connected to to VM-host-cluster-data center hierarchy, and beyond.

AUTOMATING DISCOVERY

But that just addresses the scan itself. What about subsequent scans?

A good monitoring system should also allow you to take that whole profile that we just discussed, and schedule it to run:

- Every xx hours/days/weeks
- Every x day of the week (every monday, every sunday, etc)
- Every x day of the month
- At specified times of the day

In addition, there should be controls to turn off the scan if it runs past a particular time of the day, or if it has been running for more than a set period of time.

In this way, you can break down large environments into manageable slices, set up robust, sophisticated discoveries, and not overwork both the monitoring solution itself, or the environment being scanned.

Finally, in addition to running as a scheduled job, the scan should be able to be triggered by an event. For example, if an interface on a router has been down for more than 30 minutes, start a scan on the subnet the interface was part of, to see if a new interface has been brought up, and if any new far-end devices have come online. But regardless of the triggering event, the functionality you want is to set off a controlled discovery based on real-time events on your network.

PROCESSING THE DISCOVERED DEVICES

That brings us to the question of what to do with new devices when they are found.

The trite and absolutely wrong answer is this: just add them to monitoring. Here are a few reasons why this is a bad idea:

First, because not every device on the network needs to be monitored, and even in highly regimented and controlled networks, not every device that appears in a subnet is supposed to be there. People plug in laptops to investigate issues, and leave them there overnight. Devices get plugged into the wrong ports. The exceptions go on ad nauseum.

Second, not all devices deserve to be monitored, or at least monitored at the same level. Sometimes devices are added as a hot-standby, or because a particular feature is being tested, or because the device is currently in an extended staging phase.

Third, because a device isn't just a device. It's also a collection of sub-elements, such as interfaces, disks, card modules, and even applications. A bit of review is always nice before throwing the 300-port stack-switch into monitoring and using up all your licenses.

And that takes us to my fourth point: capacity. Monitoring solutions are never un-limited. There is always a ceiling where more elements cannot be polled. This also has as much to do with the type (meaning age and speed) of the devices being monitored as it does with the raw count of monitorable items. You could have a 12-port switch that is 10 years old and responds like the proverbial tortoise. Throw a dozen of those boxes onto a monitoring polling engine, and you may find it has its hands full. On the other hand, that 300-port stack switch could be the fastest thing this side of the Millenium Falcon.

Regardless, you have to be aware of what is being added to the monitoring environment, or you will quickly find it to be overwhelmed and no good to anyone.

So, hidden within all of those negatives, we can find the seeds of what you should look for.

First, find a tool that will take newly discovered devices and list them out for approval, preferably in a report format that can be shared among teams. Second, you should be able to list out the sub-elements of those devices. Third, you should have the ability to filter certain element types out of hand. For example, nobody ever needs to monitor the CD drive for disk capacity. More to the point, these filters should be specific to device types.

Finally, your monitoring solution should ALSO have the ability to alert you when licensing or capacity is impacted by the number of devices and sub-elements being monitored. That way, if you decide to auto-add certain devices, you won't be caught unaware when someone adds 150 new switches to the network over the weekend. (True story.)

Application Discovery

While device discovery is important, it tends to be the simplest aspect of the total monitoring story. In the years since I started in IT, hardware discovery, identification, and enumeration has become fairly standardized and predictable. But applications continue to be their own ball of wax. Figuring out what is installed on a server, what is running, and what it is doing continues to be a challenge for even the most sophisticated tools.

Despite that, application monitoring vendors work diligently to create and maintain libraries of application signatures, the combination of running services, filenames in standard locations, and registry entries, that provide a high degree of certainty that software is present. There are technical challenges associated with accurately understanding what is running on a server. Meanwhile, it is critical to the business to have robust and accurate application monitoring in place, an area that demands significant attention.

So how do we ensure that the applications that are critical to our business are monitored? Here are just a few ideas:

DISCOVERY EARLY

Obviously, as soon as a new server comes on line, you should scan it for software. That means being able to kick off a device discovery based on more than just a schedule. You need to be able to scan a machine based on a trigger, such as it appearing in the monitoring system inventory.

That could be done as part of the hardware scan itself, but that's not the best strategy because when a server first comes online, it likely doesn't have much installed. Having the ability to trigger an application discovery based on an alert is a better plan. This way, you aren't hammering a brand-spanking-new server with discovery traffic when the server team is still peeling shrink-wrap off the sides. You're more likely to catch the real software on the box this way.

DISCOVER OFTEN

Just as you do with the hardware scans, you need to check machines on a regular basis to see what has been added, changed, or removed.

Sure, the server wasn't provisioned with a DHCP server, but that doesn't mean the application team might not turn it on, or that, six months later, the vendor doesn't recommend the SQL database be moved from its dedicated instance on a cluster directly onto the application server to fix ongoing performance issues. Ongoing scans will catch changes like that and ensure that you are monitoring the right applications at the right time. Make sure you can schedule these discoveries with the greatest amount of flexibility in terms of time, dates, sets of devices to scan, etc. The last thing a Server Manager in charge of 5,000 physical and virtual devices wants to hear is the monitoring engineer tell them "We start scanning for software on Sunday at 1 am. And... uh... it usually ends around Wednesday afternoon."

DISCOVER NO DEVICE BEFORE ITS TIME

Even with the controls and delay tactics described in previous sections, there's still a chance that you will scan a machine before it has the relevent bits laid down on it, and it will take time for another automated scan to pick up what was missed.

This inevitably leads to manual work for the monitoring team and/or potentially missed errors for everyone. Luckily, there's a way to avoid this: Custom Fields.

This capability provides you with the ability to skip the application scan until the server is truly provisioned. it would work something like this:

1. Monitoring is set up with a custom property called **disposition.** Choices for this field include unknown, build, burn, pre-prod, pre-prod-scanned prod, decom, etc.
2. The machine is built.
3. A hardware scan picks up the new devices.
4. A new device report is reviewed, and a server is selected for inclusion in monitoring.
5. A daily alert checks for systems with a blank disposition field. The trigger action is to set the disposition to **unknown.**
6. A daily report divides devices by vendor and model and sends out sub-lists to the owners, so the Windows servers with disposition = unknown goes to the server team, routers go to the network team, etc.
7. Those teams have the ability to view and edit those systems, adding or removing sub-elements, and updating the custom properties, in the case of our new server, to **build.**
8. The server team continues with their build activities until complete. At some point, they update the disposition field to **pre-prod.**
9. Another alert looks for devices where disposition is set to **pre-prod** and initiates an application scan. A second alert trigger action changes the disposition to **pre-prod scanned** so that the machine is not scanned more than once.

While there are more tricks to ensuring a device has the proper application monitoring, this process outlines a way in which minimal human interaction can be augmented by the automation capabilities within the monitoring tool itself.

POST DISCOVERY: TRUST, BUT VERIFY

Similar to the hardware scan results, it is important that you not just begin monitoring every application found on a server. Sure, DNS and DHCP may be present, or IIS may be running to support the application developers, but that doesn't mean you want to monitor those applications with the full force of your monitoring solution.

Like the hardware scans, your monitoring solution should allow for the results of an application scan to be displayed in a list or report, and then verified by the beneficiaries of monitoring before being added into the mix. That way you ensure a minimum of alarms on nonessential systems, while avoiding the overburdening of your monitoring solution with unnecessary components.

MONITORING ASSIGNMENTS

We've reached the part of the discussion that points out the distinction between application servers and application servers. The same application can be running on the same hardware in two very different situations and the criticality, and the components that need to be monitored, are very different. This can be due to load balancing, location within the network, business service the server is part of, whether the server is part of the production or QA environment, and more. These differentiations are essential to knowing whether to apply the critical order system MS-SQL application monitors, or the low-importance departmental MS-SQL package.

So how do you accomplish this? And more importantly, how do you accomplish this when a server could go from production to decomm status? Or start off as a QA machine before being promoted to full production later on?

The answer once again lies in those custom properties. While it may take more than one or two, it is possible to create a comprehensive system that allows your IT organization as a whole to create a profile that identifies the device for what it is. For example:

- **Net-Location:** DMZ, data center, warehouse, remote, etc.
- **Disposition:** build, stage, test, pre-prod, prod, decom, etc.
- **Business_critical:** 1 through 5
- **Primary_Use:** SQL, AD, Tomcat, Fileserver, etc.
- **Associated_Application:** email, order entry, XYZ_App, etc.

While this is going to necessarily become as complex as the application landscape itself, it allows you to set up automation to apply the correct monitors and change them as time passes.

So far, I've been strictly vendor-agnostic. And I promise that the rest of this guide is free of specific vendor capabilities, but the only way for me to describe how this type of feature works is to tell you how SolarWinds does it. If that's a turn-off for you, skip ahead to the next section to keep your agnostic sensibilities intact.

Okay, so, SolarWinds has a capability called **groups**, describing groups of devices. You can use them for a variety of purposes, including:

- Establishing the health or availability of a group of systems as a singular point
- Creating parent-child relationships so that one router is the parent of a site, so if it goes down, the rest of the devices are ignored rather than alerted as down
- Reporting on all members of a group
- Creating displays that only show members of a group

A relatively new feature is the ability to assign application monitors not to specific devices, but to a group, as defined by SolarWinds. When a device is added to the group, the application monitor is assigned. If the device leaves the group, the application monitor is removed.

Groups can be created manually by selecting specific devices, or they can be dynamically populated based on a query. You know, something like, "All devices where the disposition is **prod** and the net location is **data center** and the business_critical is 4," etc.

So as those custom fields are modified, the application monitoring will instantly and automatically adjust itself to suit the new conditions. That may mean ramping up or down on the aggressiveness of the monitors, or it could completely remove collections of monitors.

Best of all, it happens without any involvement by the monitoring specialists. And that, my friends, is the kind of automation that I'm talking about.

SHOW ME THE MONEY

Automation, especially the kind that we've been discussing here, takes time. It takes time to dream up, develop, and test; it takes time to test some more; it takes time to test one more time because we're professionals here and we know how things go; and it takes time to deploy. And, while I am sorry about the cliche, all of that time translates into money for the business.

So smart monitoring specialists needs to take the accountants into account. While this guide can't tell you everything you need to do to satisfy the number crunchers at your company, this list of suggestions should at least get you started

Remember the Bad Old Days?

You need to, at least from a numbers perspective. Make sure you have data on the ticket counts before you put new automation in place, so you can say something like, "Before monitoring, we were generating 800 systems-related tickets a month, with approximately 200 for interface issues, 400 for system outages, 150 for service failures, and 250 for assorted application issues. After implementing automation, those tickets dropped to, etc..."

Avoid the Retro Encabulator

Watch this video. This, my friend, is what you sound like to everyone else. It's especially what you sound like to the business leaders at your company. They may nod appreciably, but there's a good chance that your pet iguana understands more of your conversation than they do.

So, in the name of getting what you want, skip it. All of it. Instead, put things into terms they care about. You got the ticket data from the previous step, right? Now turn it into dollars.

The algorithm isn't difficult. Set an estimated amount of time to resolve each of the ticket types that you listed in the previous steps. Something like:

- Interface issue – .75 hour
- System outage – 1 hour
- Service failure – 1.5 hour, and so on

Now ask your number crunchers what the average total loaded cost of an employee is. This is the hourly rate for an employee that factors in everything about them, including their portion of the heat and electrical bill.

Now multiple the <total loaded cost> x <time per issue> x number of tickets for that issue.

Now you know the total cost for that issue for that period of time. If you do this for both the before and after phases, you know how much you've saved the company, just by adding automation.

To help you out, I've included a whole section of real-life examples near the end of this document.

Gather Your Posse

Despite our suspicions to the contrary, your typical business leader's day does not comprise three hours spent skimming ***The Wall Street Journal,*** followed by a round of golf, followed by martinis, followed by attending one meeting (with you), listening perfunctorily, and saying "no" at the end.

In fact, it's likely that their day is an endless series of meetings where the person at the front of the room shows off how much they know (c.f. "The Retro-encabulator" in the previous section) and then saying some version of "trust me, this is important", followed by a request for money that the business leader suspects is 3 times what is needed to protect against eventual budget cuts.

One of the best ways to attract the kind of attention you want from important decision-makers in your company is to provide solid numbers. Another way is to show your results in the form of end-user testimonials. When coupled with numbers, positive descriptions of experences before and after implementing automation provided by people from various departments reinforce the idea that the not-trivial effort required to implement automation is worth the investment.

Doing this also has the secondary benefit of reinforcing the message that investing in the monitoring solution was also a good decision, and that further investments will have similar positive outcomes.

Revenue, Cost, Risk

During the 2015 THWACKcamp session, **"Buy Me a Pony,"** I discussed the drivers that help executives make decisions with the SolarWinds CTO. He boiled it down to three things:

- Increasing revenue
- Reducing cost
- Avoiding risk

If your project, software, initiative, etc., can't speak to one of those things, it's simply not going to be a priority for them.

The good news is that monitoring automation does at least two things:

Monitoring reduces costs – catching issues sooner in the failure process, potentially before they spiral out of control, clearly helps reduce business expenses. As we've already discussed, automation takes that a step further by taking action at the time of the event in an attempt to avoid the issue entirely. All of that translates to reducing the cost associated with the avoided impact of the issue.

Monitoring helps avoid risk – conversations about risk mitigation and avoidance frequently focus on ways for teams to predict, and then circumvent, potential failures that could affect a system, application, or service. Monitoring and automation come into play when you've come down from that philosophical mountain and accepted that some failures are simply unavoidable. So in this case, the risk automation helps you avoid is the downstream consequences of the failure, by detecting it faster and responding to it as soon as possible.

THE PLURAL OF "ANECDOTE"

If you are just getting started with automation, it may still be hard to imagine that a simple script could have that big an impact on your environment. To help with that I've collected a few "war stories" from friends and colleagues. Use these as you start to socialize the idea of monitoring automation in your company, at least until you have stories of your own to tell.

It's important to understand that I didn't include these stories because the automation was particularly complex or the savings were in the stratosphere. Quite the opposite, in fact. The solutions featured here are extremely modest and easy to accomplish, and the savings are meaningful but not millions. Trust me when I say that, once you get started with monitoring automation those million-dollar opportunities will present themselves. But the lesson here is that smaller efforts still yield measurable returns.

Service Monitoring Starts at Home

Josh Biggley, Monitoring Engineer for Cardinal Health, shared this with us:

"As our monitoring is heavily dependent on SNMP, we have an alert in place that checks whether that service is actually responding on our servers. If not, we automatically restart the service. From its inception we were auto-restarting the SNMP service and created an average of 1.98 incidents per day. In Oct 2014 tried removing the logic to restart SNMP. Between Oct 11th 2014 and Jan 12th 2015 we averaged 6.35 incidents per day, an increase of 4.37 incidents per day. Even if the total incident handling time was only 15 minutes for all parties involved, removing that logic added an extra 65.5 minutes of work per day or nearly 400 hours per year. That's 1/5th of an FTE. As you can imagine we decided to re-enabled the SNMP restart logic."

Clearing TEMP-tation

Also from Josh Biggley, Monitoring Engineer at Cardinal Health:

"While we developed some very sophisticated alerts for "disk full" monitoring, we still were creating over 700 tickets per month for this one event type – the largest volume of tickets for a single event in the enterprise. As we discussed it with the server teams, they pointed out that in the majority of cases clearing the temp directory was all they had to do to resolve the issue. So we collaborated to test and deploy a solution to do that automatically. To give you a sense of how successful that automation is for us, we deferred 408 tickets between August 8 and August 31. We presume that responding to a "disk full" event takes staff an average of 15 minutes to manage. But even with that minimal time, we've still saved 17.7 events, or 4.4 hours per day. Automation is saving us 1/2 of a support engineer each and every day for just this one event type !"

Flagging the Problem Children, Then Expelling Them From School

Peter Monaghan, CBCP, SCP, ITIL ver.3

"We've setup similar scripting actions regarding diskspace alerts and Windows Services unexpected stops. Since we don't have a 24x-7 NOC we have introduced "repeated" SolarWinds alerts for Tier 1 IT Services so that if one of those services fails afterhours scripts are initiated upon alert generation to automatically restore services. If services aren't restored alerts are generated every 30-45 minutes (depending on service) until either the script or manual intervention restores it.

Once I setup monitoring for these services I was able to focus on the repeat offenders and ultimately reduce all chronic alerts and outages by over 70% for all Tier 1-3 IT Services."

Signal to Noise

Rick Schroeder, Network Administrator, SCP & THWACK MVP

"We experience approximately four unscheduled WAN outages per month. Our affected sites range from five employees to three hundred.

Using monitoring automation we reduced the amount of down time by fifteen to sixty minutes per site, per incident simply through increased visibility by getting the right information to the right teams faster, so they could respond. That included providing near-real-time information on outages to the WAN vendor so they could begin verifying and testing, perhaps bringing in last-mile-providers, intermediate providers, or rolling a truck & technician to the site.

Some of the outages were six hours or more – if the issue was caused by backhoe fade or trees falling on aerial WAN connections.

Depending on the number of users affected, we calculated our savings at $86,400 per year for our larger sites, and $375,000 for outages at our data center (which only occurred once every 3 years or so, but in addition to being costly, it's also very visible).

And all of that doesn't consider the intangible costs for customers lost and impressions left."

Repetitive Strain

Will Luther, Analyst Network Operations, GVTC

"A few years back before we had any automation in place, I had to manually go through and backup the configs for all of our devices (started with 200+)... So, every week, I would manually connect to EVERY device, paste my commands in, and backup the configs. This process took me approximately 2 full workdays to complete... EVERY WEEK.

As our network grew, my notepad of commands to be copied and pasted, evolved into a collection of "expect" scripts. This was a major leap forward, and cut down the time to backup significantly. While it had previously taken *me, roughly 2 full days to perform all of the device backups, I was now able to backup those same devices in less than a day.*

Our network eventually grew to be 1000+ devices, and even with those expect scripts, it started to take longer, yet again. Additionally, in this time frame, we had several devices die. And, while I was able to eventually provide the backed up configs, it was still cumbersome and time consuming.

I finally got the approval to get an automated configuration management system

That was the answer, and the last evolution of our config management system. Now, I spend exactly 100% of 0 days each week, backing up configs. The configs are easy to find, which makes re-configuring devices painless.

So automation has recovered 824 hours of staff time per year, not counting the faster resolutions when there is a device failure."

Nobody Wants to Hear "Oops" in the Operating Room

Charles Hunt, Network Monitoring Specialist

"At one point the Hospital I worked at had an in-house developed EMR system which was distributed across several servers. The system would fail if a single service on any of those servers stopped unexpectedly. That application team's way of managing this was logging each server to check the service/process status one at a time until they found the stopped or hung service and then restart the servers in a specific order that would allow the service/process to restart and the servers to reconnect with each other and bring the EMR application back on line.

Most outages averaged 1-2 hours and required the team of 4-6 developers ($50-80/hr) to search each server for that one service/process.

Implementing monitoring alone saved the company no less than $200 per outage just for the development team's time, let alone the cost of the entire hospital staff whose work was held up.

Also a factor was that reducing the down time from 1 hour to 30 minutes meant we only missed/delayed one appointment instead of 2 or 3.The impact on delayed surgeries was even more noticeable."

The Rx is Restarting Services

Charles Hunt, Network Monitoring Specialist

"The Rx Group agreed to monitor the few specific services that were critical and cause the robots that dispensed medication to shut down.

With simple notification, outages went from an average of 1-1.5 hours to 20-25 minutes and translated to a savings of $266 in staff time per event.

On the business side not having an extended delay in the robots dispersing medications was (literally) invaluable to the medical staff and patients when you consider the risk to health and possible lawsuits."

The Right Information at the Right Time

Paul Guido, Systems Team, Regional Bank in south Texas

"We monitor approximately 100 data circuits. According to our records, a circuit goes out about fifty times a year due to carrier issues, floods, dry spells, drunk drivers, ice storms and (everybody's favorite) backhoes.

Before monitoring we would have to wait from thirty to one-hundred-twenty minutes before IT knew about the issue. To see if the issue was on our side a person would be dispatched to the site. A typical drive time would be an hour. If the smart jack shows an error on the carrier side, a ticket would then be opened manually.

In addition, circuit outages cost a branch operation six to eight additional man hours per circuit incident.

After implementing monitoring automation, a ticket is opened with the proper carrier within ten minutes. Our company and the carrier investigate the outage to see what is at fault (us or them). Because all the required information is gathered and included in the ticket, every hour of down time only causes three to four man hours lost at the branch.

In a single year, circuit monitoring saves 250 to 400 man hours of productivity."

The Price of a Cup of Coffee

Kimberly, SysAdmin

"We have an application that approximately 470 developers use to drive their development cycle. The application would occasionally hang during a reindexing over night and wouldn't be available when the majority of the devs arrived at work between 7-8am.

I'm more of a 9 o'clocker myself, and so I'd either get a call during my not so awake moments or they would have to wait for me to arrive in the office. Our devs make about 33.5 an hour.

Prior to automation, the math looked like this: 470 developers down for 2 hours at 15,755 per hour, or approximatly 31,508 in lost employee productivity.

I was able to set up a monitor for that reindexing job and when it hung, executed a script to which restarted the service and sent a message to our operations center to look at the app, confirming that it was accessible after the restart.

From that point on, the devs could start to work right away, the company didn't lose employee productivity, and I got to drink my first cup of coffee uninterrupted.

Truly priceless!"

The Little Things Add Up

Jason Higgins, Network Analyst

"We have an application/service on a print server that allows our billers to print remotely stored data locally through encrypted channels. This application/ service would frequently (2-6 times a week) hang up and stop responding on the server. The fix is simple enough, just remote into the server, find the service, and restart it, and took about 10 min total from call to resolution.

The problem comes in when people don't report the problem all day, and then call in right at the end of the day in a panic. Our help desk is only three people, and they are frequently not at their desks because of being out working on *problems. The lead person for the week carries a side phone to take calls on the go, but even then you have to find a computer to get on and so on and so forth.*

I setup a job to monitor that service and alert when it was not running or not responding. This was a great first step because the help desk could see the alert, remote into the server, restart the service. Usually this took place before the end users even knew it wasn't working.

But again this went on anywhere between 2-6 times a week. After having the alert setup for a few weeks, I discovered I could go a step further and actually restart the service when it stopped responding. I put this into place, and now the help desk does not even have to do anything when the error happens, it takes care of itself, and they just log the ticket.

We calculated the cost of a single outage, in terms of staff cost, was just $16.31. But this is one of those cases where the math adds up... 2-6 incidents per week means $32-97 per week, which adds up to $1,696 to $5,044 per year. Maybe not a big deal to some larger companies, but it was certainly a big deal to us."

THE COMPLETELY UNNECESSARY SUMMARY

Good automation is enabled by, and is a result of, good monitoring. When done correctly, it is elegantly simple, and most importantly, it's not artisanal—it's just automation the way automation is meant to work.

There are other examples of monitoring automation than the ones we've provided in this guide, but what I want you to leave with is the understanding that the biggest barrier to implementation at most companies is not the wrong tools, or the wrong skills. It's having the wrong mindset, one that says monitoring and automation are complicated and difficult, *"Far beyond the ken of mortal man,"* to quote the old Superman reruns.

In the end, monitoring and automation are limited only by your ability to imagine and implement, assuming you've got a good monitoring tool in place, rather than your ability to perform some weird interpretive dance.

DEDICATIONS

To Debbie

You have been there – by my side, as a friend, as the most trusted advisor, in my life for so long I can no longer tell where you cease and I start. I love that this part of our journey lets me be home with you so much, and that you patiently listen to me explain, gush, rant, and share. I love you F, E, & A, best beloved.

To the SolarWinds Head Geeks

You are four of the most incredible, talented, exciting people to be around. Having you there to bounce ideas off, to get excited about video ideas with, to pass geeky movie clips to, these are all blessings that go far beyond the amazing job we share. Thank you for letting me be in the cool kids' club.

solarwinds