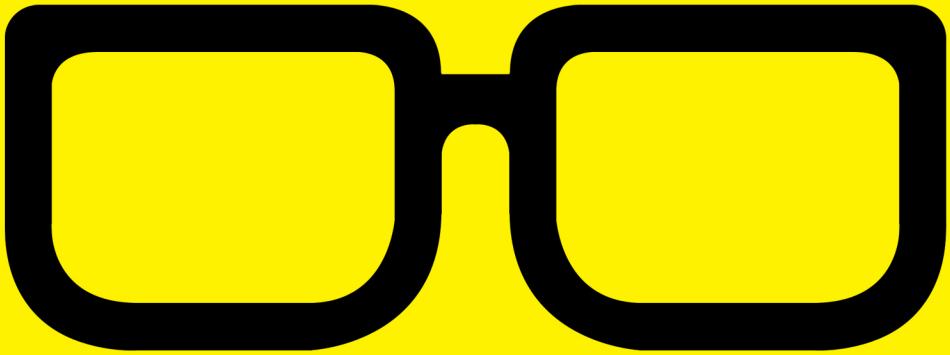
SPONSORED BY

GEEK GUIDE



Improve Business Processes with an Enterprise Job Scheduler

LINUX™ JOURNAL

Table of Contents

MIKE DIEHL has been using Linux since the days when Slackware came on 14 5.25" floppy disks and installed kernel version 0.83. He has built and managed several servers configured with either hardware or software RAID storage under Linux, and he has hands-on experience with both the VMware and KVM virtual machine architectures. Mike has written numerous articles for *Linux Journal* on a broad range of subjects, and he has a Bachelor's degree in Mathematics with a minor in Computer Science. He lives in Blythewood, South Carolina, with his wife and four sons.

GEEK GUIDES:

Mission-critical information for the most technical people on the planet.

About the Sponsor

Skybot, A Division of HelpSystems

HelpSystems has more than 30 years of experience in providing enterprise scheduling and automation solutions. Part of the HelpSystems family of brands, Skybot Scheduler provides a flexible, easy-to-use solution for cross-platform enterprise job scheduling, allowing businesses to integrate workflows across servers and critical business applications and monitor them from a central interface. Skybot also includes robust reporting, auditing, and security capabilities to ensure that your enterprise job schedule is well documented and reliable.

For more information on Skybot Scheduler, see www.helpsystems.com/skybot.

Improve Business Processes with an Enterprise Job Scheduler

MIKE DIEHL

Introduction

Modern IT shops run a whole lot more than just a few file and print servers like they did in the old days. Today's enterprises are vastly more complex, often with servers in different data centers and even scattered all over the globe. Additionally, they typically have Windows servers, Linux servers, mainframe servers and various other flavors of

UNIX servers, and all of these servers, all over the network, produce and process data.

As a broad example, in many enterprises, sales departments produce data in the form of, well, sales. This sales data somehow has to cause shipping or fulfillment departments to ship orders, and that may require something to happen in warehouse or procurement departments. Of course, then the accounting and billing departments all will need the data in order to do what they do as well. And finally, business intelligence tools need this data in order to forecast sales and optimize product selections.

As you might imagine, no one tool fulfills all of the requirements of all of those different departments. Most of those departments have their own specialized software tools that allow them to do what they need to do in an efficient and intuitive manner. The problem, of course, is that all of these silos need to talk to one another so the organization as a whole can function properly and efficiently.

To make matters worse, many of these departmental tools likely run on different types of servers and different operating systems. Although each tool, by itself, may run just fine on its own architecture, having a variety of architectures can make job coordination and monitoring quite a challenge.

In the old days, all of this communication between the silos mentioned earlier would be done with printed forms and reports. Although this mechanism effectively masked the sources of the data, these

However, PowerCenter doesn't have a particularly robust job scheduler. This means you don't have the flexibility as to how and when you schedule your ETL-based business processes.

"communications" were prone to being misread, misprinted and misinterpreted. Thankfully, those days are over for most organizations. With the advent of ETL tools, people can *extract* data from one silo, *translate* it into a useful format and *load* it into another silo.

Informatica's PowerCenter

Informatica's PowerCenter is a very popular ETL (Extract, Transform and Load) tool that can collect data from a wide range of sources. The comment I made earlier about how departmental tools do their particular jobs well also holds true for Informatica's PowerCenter; it does ETL very well.

However, PowerCenter doesn't have a particularly robust job scheduler. This means you don't have the flexibility as to how and when you schedule your ETL-based business processes. For example, you might want to schedule a job to start as soon as a data file appears on an FTP site, or you might want to schedule a job to run after multiple prerequisite jobs have

completed. PowerCenter isn't going to provide you with that type of flexibility.

Because Informatica's PowerCenter isn't meant to be a sophisticated job scheduler, you also may find yourself having to intervene once in a while. This might happen if one job fails and then causes dependent jobs also to fail. Or, a time-critical task may be held up until the next scheduled polling cycle, just because a single file wasn't uploaded in time. Of course, there are ways to automate a response to any of those types of failures, but at that point, you're probably writing batch files or shell scripts to work around an inherent weakness in PowerCenter. There's a better way, as you'll see.

To make matters even worse, many of the jobs upon which PowerCenter relies will be run on other servers in

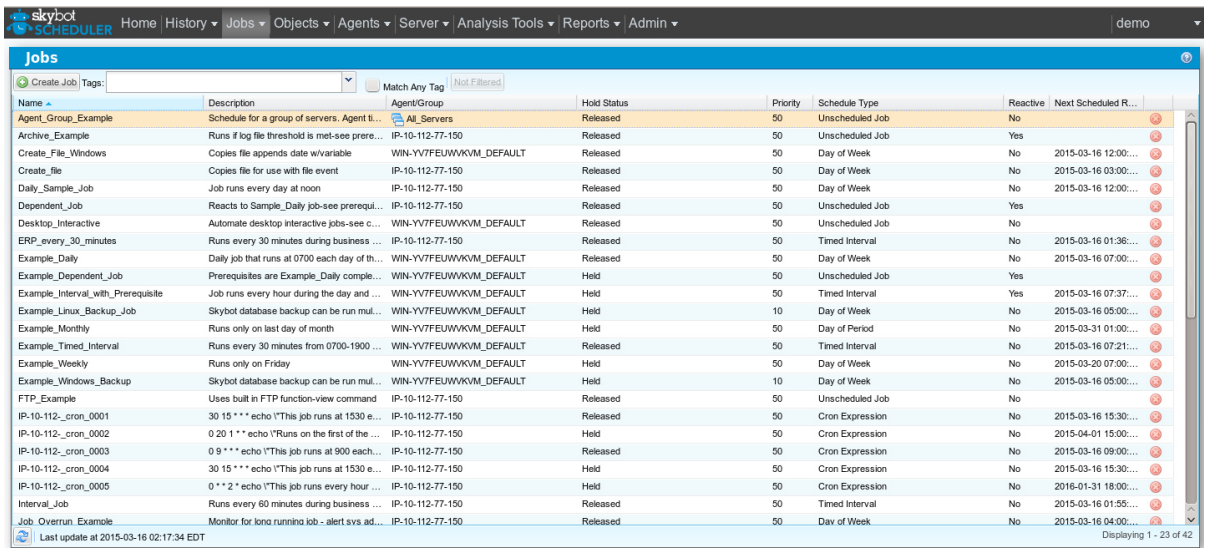


FIGURE 1. Flexible job scheduling is a key ingredient for an efficient ETL operation. Skybot from HelpSystems is a world-class enterprise job scheduler.

This dilemma points out another often overlooked issue with distributed ETL in general. There is no way to know how things are running—until they aren't.

an enterprise. This means that those jobs probably will be scheduled on whatever servers on which they run. So now, in addition to not having a sophisticated job scheduling mechanism, in this scenario, scheduling has been decentralized by the use of multiple scheduling tools. This quickly becomes difficult, if not impossible, to manage.

So here's the question: "Now that you have all of this running, how do you monitor it from day to day?"

Monitoring

Depending on the size of your enterprise, you could have a huge, tangled web of interconnected servers and interdependent jobs. How do you summarize the general health of such a process? What if something goes wrong? Who discovers the problem? Who gets notified, and by whom?

This dilemma points out another often overlooked issue with distributed ETL in general. There is no way to know how things are running—until they aren't. Sure, it's basic monitoring, but again, Informatica's PowerCenter doesn't do monitoring; it does ETL.

Goals

The purpose of this ebook is to point out some pain points you probably already know you have and show you what you can do about them. By understanding the deficiencies outlined previously and using the right tools to mitigate these issues, the goal is to help with the following:

1. Save you money on resources and staffing requirements.

2. Help you build an efficient and reliable business process.

3. Show you how to build new processes in a repeatable fashion.

4. Improve your quality of life—yes, really.

Let's see how.

Enterprise Job Scheduler Benefits

Centralized Command: A true enterprise-grade job scheduler will provide a centralized command and control capability, and this is crucial. By centralizing job scheduling, you no longer have to manage multiple scheduling systems. Jobs on Windows, UNIX, Linux and mainframe systems can be scheduled from one central location, using a unified interface. And once it's all set up, you don't even have to keep track of details like where a job is supposed to run or on which operating system it runs.

An enterprise-grade job scheduler can schedule jobs based on things other than just day of week and time of day.

This centralized, architecture-agnostic approach opens up another opportunity. All of the different locations and departments within an organization that previously had been apprehensive about "joining the fold" may be more willing to allow their jobs and processes to be managed by a core IT group. This will be especially true once the various groups understand that the authority to manage their critical processes still can be delegated back to them. They should appreciate the fact that they don't have to give up control, but still can gain centralized process support.

Scheduling beyond ETL: If you've made the investment into an ETL tool like Informatica's PowerCenter, you've no doubt tried to find every data-related process that can be automated using the ETL tools available. The same thing is going to happen once you invest in an enterprise-grade job scheduling tool. You will rediscover that other jobs, besides ETL, can be scheduled. Obviously, almost anything that runs on a server can be scheduled, but suddenly, server backups, log rotations, audit reports and other such jobs all can be scheduled across the entire enterprise from a central command and control center.

Flexibility: An enterprise-grade job scheduler can schedule jobs based on things other than just day

of week and time of day. For example, a job can be scheduled to process a data file as soon as it is uploaded to an FTP site or when a VPN connection becomes available. This could cause a paradigm shift in how you think about scheduling work. Instead of periodically polling to see if anything needs to be done (and having to worry about what happens if some part of the process misses a deadline), you now can perform a given task as soon as it's ready to be performed.

This also means your jobs can be on a tighter schedule since there no longer will be any need to cushion certain jobs because prerequisite jobs are expected to finish within a specific window of time. Simply schedule jobs as dependencies, and they'll run back to back. Being able to run jobs on a tighter schedule could have positive staffing ramifications if you have technicians monitoring the processes, because processes that run faster don't need to be monitored for as long.

Simplification: Even more intriguing is the fact that you no longer will need to write potentially complex checks to see if previous jobs completed before starting the next job. You simply would create the other jobs as prerequisites for the final job and let the scheduler keep track of what has run, what failed and what still needs to run before a given job runs. Jobs that fail also should have an automated response to handle that failure, even if that response is simply to contact an on-call technician to deal with the situation. Notifying an on-call technician once in a while is certainly preferable to maintaining on-site staff to monitor business processes in real time.

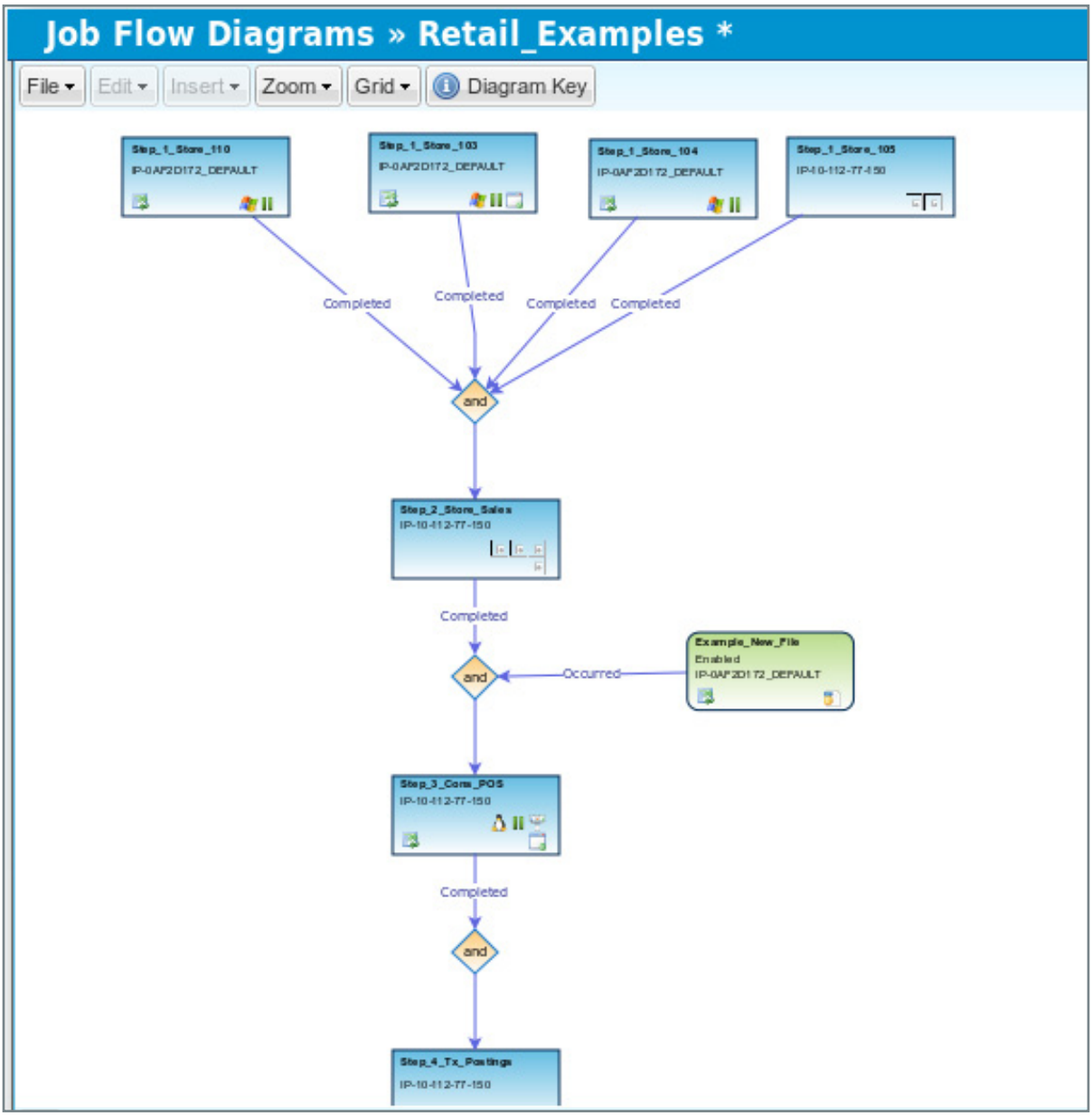


FIGURE 2. Complex job dependencies are easy to manage with the right tools.

Scheduling flexibility is a great thing, but it's not so great if it's difficult to use. The reality is that what you're doing now is probably more difficult—employing

different schedulers, with their own idiosyncrasies, for each architecture in your enterprise. Windows scheduler and cron for UNIX and Linux aren't difficult to use, but that's still more complexity than you need and they don't handle job dependencies. A single, enterprise-grade, job scheduler presents one user interface and the ability to schedule jobs no matter where they need to be run. This means less training burden and a more efficient operation overall.

IT Dashboard: This idea of a unified job scheduling interface leads to another concept that I've been hinting about throughout this ebook so far, and that's the concept of absolute situational awareness. You simply need to know how well things are operating

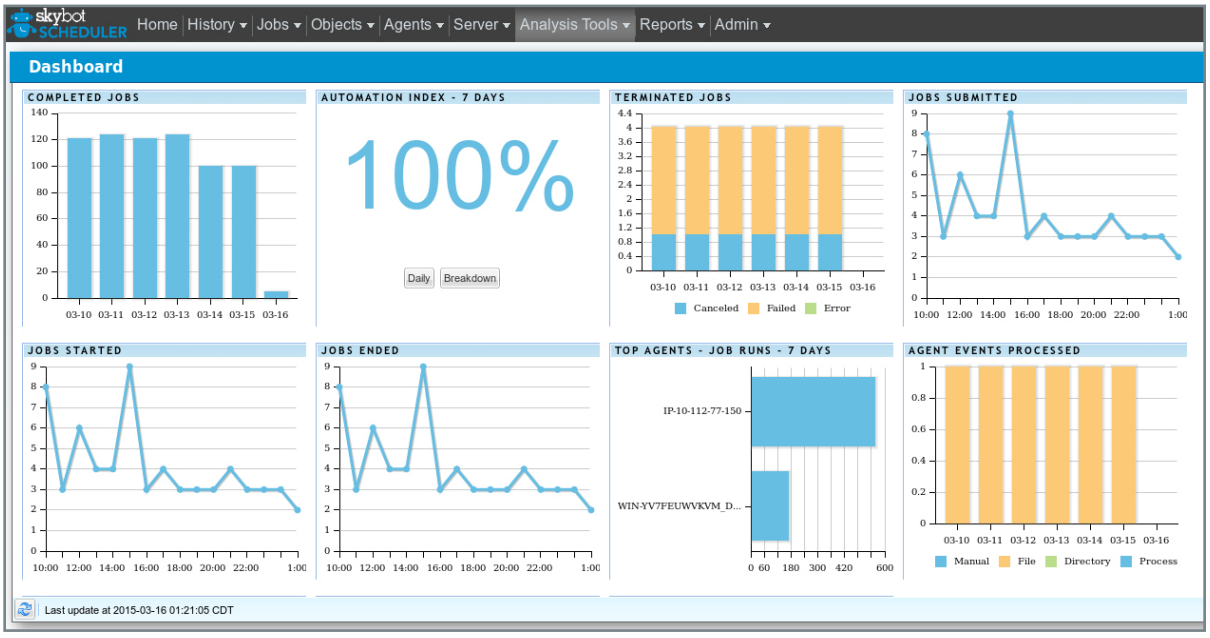


FIGURE 3. **Process monitoring should become part of your IT dashboard.**

The most important requirement for your IT dashboard is that it presents the current state of affairs in an easy-to-use, intuitive manner.

from a process point of view. That is, are all of the ETL jobs running? The backups? The purchasing reports? How about the business intelligence models that your manager looks at only once a week but expects them to be there just the same? A centralized job scheduler, if it provides sufficient reporting and notification functions, should become part of your IT dashboard.

The other important part of your IT dashboard is your server and network monitoring system, but it's important to remember that network monitoring is a separate function from process monitoring. And once again, use your tools for what they do best. Network monitoring should be done by a network monitoring tool, just as process monitoring is best done by the tool that schedules the processes.

In the interest of completeness, your IT dashboard also may need to include physical, server and network security monitoring, but that is beyond the scope of discussion here.

The most important requirement for your IT dashboard is that it presents the current state of affairs in an

easy-to-use, intuitive manner. If you need to find out whether a particular job ran last night, you shouldn't have to go hunting for that information; it should be right at your fingertips. Eventually, you should be able to adopt a "management by exception" policy so that the information displayed on your IT dashboard are only items that actually require administrative intervention.

Reports: The final aspect of absolute situational awareness that warrants at least a brief mention is the ability to generate a "morning news report", outlining what jobs ran the previous day, which ones failed and how long the successful jobs took to complete. At the very least, this report should be suitable for distribution to management. This way, management has the status information it needs without having to request it from the IT department. This saves time for everyone involved. By knowing how long various jobs take to complete, you may be in a position to spot trends and predict problems proactively.

Delegation: The last element of a job scheduler that you need in order to accomplish the goals mentioned earlier is the ability to delegate authority and responsibility. Let's face it, system administrators are busy. They don't need to be too involved in the day-to-day monitoring and scheduling of business processes. That task should be delegated to lower-level IT staff members. Also, you should consider allowing department managers to be able to at least see the status of the jobs that affect their department. You might consider delegating job scheduling duties to the

more savvy members of each department. For example, let the sales department manage jobs that affect the sales department. This takes some of the burden off of the core IT group and allows different departments the flexibility to do what they need to do efficiently.

Conclusion

Earlier, I mentioned four goals. Let's see how the enterprise job scheduler has done.

By efficiently scheduling jobs, the hope is to remove processing bottlenecks. This, in turn, should extend the useful lifetime of your existing infrastructure. Since the job scheduler is able to manage dependencies and job failures, you won't be as dependent upon staff to monitor business processes and respond to any issues that may arise. Also, instead of having to staff the operations center fully, while business operations are being performed, off-hours staff can be reduced, as much of the work they would need to do would be automated by the job scheduler. Deploying a true enterprise-grade job scheduler should save your organization money.

The next goal was to help build an efficient and reliable business processing infrastructure. Suffice it to say that much of the money savings mentioned earlier comes from running a more efficient operation. But, with the flexibility that a dedicated job scheduler provides, you'll be automating tasks that may have been run manually in the past because it just didn't seem to be worth the effort to automate them before.

Additionally, the job scheduler keeps track of job failures and can notify technicians to resolve any issues, adding efficiency and reliability to your operation.

Obviously, I've been focusing on automation, so process repeatability is a pretty easy goal to meet. But, because of the centralized job scheduler, you would be able to create "job templates" that then can be used whenever a similar job has to be scheduled or even for one-off jobs that are run only occasionally. For example, you might create a generic "log rotation and analysis" job and then schedule it to run on all of the Linux servers in your enterprise.

The last goal was a lofty one—that of an improved standard of living. No, none of these improvements are going to make you rich or more attractive—sorry. However, the first goal of saving money contributes to an improved standard of living, since saving the company money often translates to job security and other forms of compensation. The concept of absolute situational awareness outlined earlier means not coming into work in the morning to find a line of people waiting at your desk because something went wrong the night before and no one knew about it. Being able to leave work at the end of the regular workday, or earlier, because you know that you'll be called if something goes wrong and needs your attention is nice. For many of you, that probably will be an improvement in your current standard of living.

Adding a job scheduler to the enterprise may seem like a lot of effort with little return. After all, your

enterprise is running just fine as it is, right? However, to summarize what I've discussed here, you're probably having to deal with multiple servers and perhaps even multiple scheduling architectures. Because of the distributed nature of this architecture, it can be difficult to determine if a particular job ran—until the phone calls start coming in indicating that it didn't. You're also having to keep track of where particular jobs run. You may not actually have a spreadsheet that documents where jobs run, but somehow, you are having to keep track of this information. Many of you are having to write your own code to verify that preconditions have been met before various jobs run. With a true job scheduler, that effort becomes almost unnecessary. Those manual checks may not be all that difficult to write, but they do add complexity to the system in a manner that is difficult to document and maintain.

Finally, an efficiently running operation requires fewer staff to monitor and maintain. This can either save money in the form of reduced staff requirements or save time by enabling existing staff to focus their efforts on more important tasks.

Many enterprise-grade job schedulers exist on the market today, but I invite you to take a look at Skybot Scheduler from HelpSystems and perhaps use it as a baseline for comparison to other products. You can get more information about Skybot at http://www.helpsystems.com/skybot.■