

Job Monitoring

*Job Monitoring: System Informer or Sixth Sense?
Why employing the biggest snitch and know-it-all
is good for business*

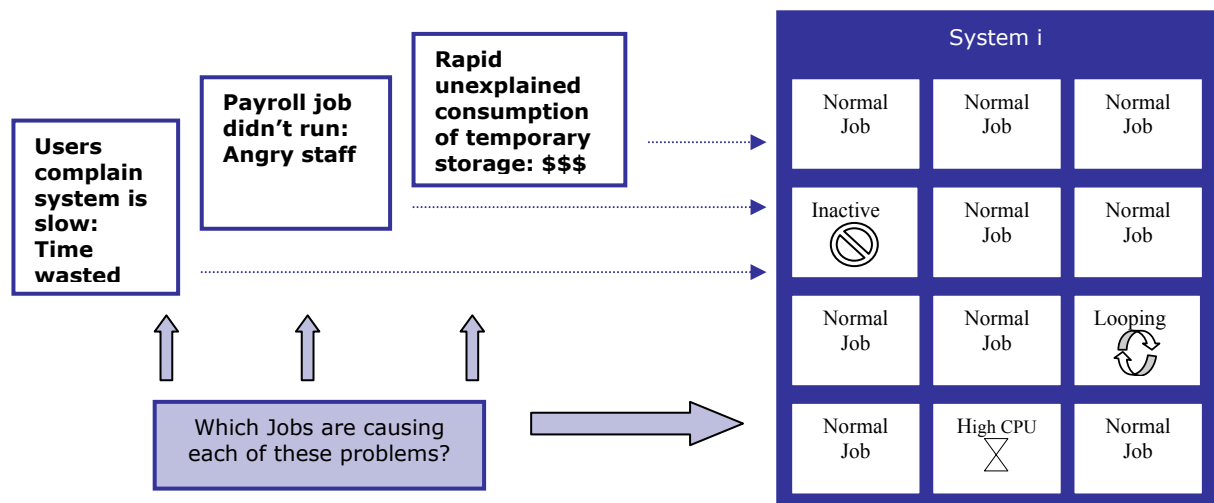
Job Monitoring – A Friend on the Inside

Imagine you had a friend that turned up right before something bad was going to happen and told you how to avoid it. It could be some small incident involving your shoe and a sidewalk nasty, leading to flared nostrils and suspicious looks at the office all day. Or it could be something much worse, like if you don't attend to that looping job silently going about its business, the system in India will fall over and the downtime will cost the company hundreds of thousands of dollars in idle work hours. Yes, a friendly forewarning wouldn't be bad in either situation. For the first, we suggest you watch where you tread and if that doesn't avoid the inevitable, flare your own nostrils with empathetic disgust and throw a silent nod of blame to the work experience student. For the second, we would suggest employing a job monitor – it's your friend on the inside.

Rebel with a Cause

At the user level, the systems powering the business applications need to be the ever available, fully optimal, always-on magic behind the scenes that demands nothing of the user and is devoted to a life of service, duly processing jobs efficiently, with a minimum of fuss. As most System Managers would agree, when this utopian vision starts to fray around the edges, a fair amount of finger pointing is levied at the system as the 'cause' of issues that are felt directly on the user level, regardless of the user actions that may or may not have contributed. The system, as such, is not necessarily to blame, it is only guilty of not controlling the rebellious jobs in its charge. Rebel jobs, when provoked, have the potential to do things they should not. Rebel jobs loop, become inactive or gorge themselves on temporary storage. Rebel jobs gang up to cause mischief and hide out in obscure subsystems where they can avoid detection.

System Managers may be well aware of the chaos rebel jobs cause, the evidence of their rampage can be seen in user complaints, important jobs that do not run or resources that are being consumed. The challenge for them is to get the inside track on how to identify and round up the trouble makers before the damage is done. When faced with a system or network containing hundreds of thousands of jobs, this can become a task of overwhelming proportions – in terms of time and money.



Calculating the Cost

A system without an efficient means of identifying and resolving problematic jobs shifts the burden of these tasks onto what is likely an already stretched team. The team is forced to respond to problems as they occur and sift through the masses of jobs to find the culprit and ultimately, resolve the issue. Jobs that go awry can so readily impact the system environment around them. Anomalies in these environmental conditions or in routine processes involving the job often can be the first sign of job issues, that left unchecked or unattended, add up to a significant financial loss over an annual period. Two essential considerations for job monitoring include:

- Job Performance
- Job Status

Case Study

Company x is a large Financial Services organisation that is struggling with job issues on their centrally managed System i network. The network supports 21,000 users nationwide and the company generates \$4.2 billion in revenues annually. In a review to outline the extent of this issue, they calculated the associated financial impact over the previous year's operations. The cumulative effect of these problems throughout the network and the impact on users' productive time was far more than they originally thought.

Area	Problem	Consequence	Cost
Job Performance	A looping job is consuming vast amounts of temporary storage	Time consuming investigation and additional disk is required to avert system crash	\$2,700 ¹ Annual Est: \$13,700
Job Performance	Consistently high CPU usage as several QZDASOINIT jobs take more CPU than they should	Time consuming, jobs all have same name, which ones are causing problem, meanwhile user productivity slows	\$225,000 ² Annual Est: \$1,800,000
Job Performance	Poor performance of certain jobs due to a high rate of faulting in a specific pool indicating too many jobs in the pool or insufficient memory in the pool	Time consuming, which memory pool are jobs running in, then how many jobs are in pool etc.	\$24,000 ³ Annual Est: \$96,000
Job Status	An important accounts related job becomes inactive but this is not detected immediately	Users are delayed in their work waiting for the large job to be processed	\$1,800 ⁴ Annual Est: \$9,000
Total Annual Cost of Job Performance and Status issues			\$1,918,700

Key:

- ¹ Six 5hr incidences of wasted profitability hrs for 3 people and 3 idle workers' time and extra disk
² Eight 6hr incidences affecting a group of 1,500 users running at ½ productivity capacity
³ Four 3hr incidences affecting a group of 320 users running at ¼ productivity capacity
⁴ Five 2 hour delays affect a group of 30 workers running at zero productivity capacity

The Usual Suspects – Users Vs Jobs Pt 1

Whilst systems differ enormously from company to company, there are a few jobs that seem to attract problems and utilise vast amounts of resource until they are detected, regardless of the system set up. Managers will be well aware of them and the legacy of trouble they have caused. If such jobs are well known, then it is a significant benefit to attach appropriate automation on the back of identifying a problem job. CCSS has designed a specific job check feature for just such occasions. MONCHKJCP runs in the background and keeps an eye on potentially problematic jobs (or jobs that would cause substantial issues if problems went undetected). This command checks the CPU usage of the job and then takes the appropriate action, e.g. hold, lower its priority or take no action. In this case, Managers have the option to configure the level of CPU usage at which action will be taken. To specify further, Managers can include or exclude generics and users within the check.

A good example of this check saving managers endless hours of investigation is when a user logs on to the iSeries then switches off the PC, believing he is 'logged out'. The user may have entered the iSeries under the generic QUSER profile and as there could be potentially hundreds of QUSER's on any given system (let alone network), the task to hunt down the particular culprit that is consuming resource becomes a painfully drawn out and expensive process. In this example, Managers would simply have to set up the check to monitor all QUSER's in a particular subsystem and if the problem arises, actions will be automatically taken without resource impacting further.

Job Status - Users Vs Jobs Pt 2

Another common example of users getting caught up in the issue of jobs going awry concerns the job status. In the course of regular processing, a user may run an important job under the wrong profile. The result could be that the profile does not have the correct level of authorisation to complete the job and subsequently, the job fails and the subsystem ends, making it unavailable to other users. When the job in question is an important one, such as 'End of Day', the full impact of this sequence of events can start on a Monday afternoon and not be resolved until Tuesday, when irate users start a flurry of calls to the helpdesk to ask why they can't access the work they need to.

Issues of Perception – Users Vs Jobs Pt 3

There can be instances where diagnosing the underlying job problem becomes an exercise in eliminating possibilities. These scenarios happen when the user perception is the system is slow (response time) but managers are unable to easily see why as there appears to be no significant performance issue. A relevant example can be seen in the 'Lock Wait' scenario. Here, a user may be experiencing substantial delays as the application program they are running is trying to access an object that is in use by another job. It 'feels' like a delay but in fact, they are locked out. It would be very difficult to diagnose the lock wait status without a dedicated monitor to supply the

average lock wait time per transaction for interactive users. In this case, if average times are exceeded, managers could be immediately alerted before users call the helpdesk.

Human Error – Users Vs Jobs Pt 4

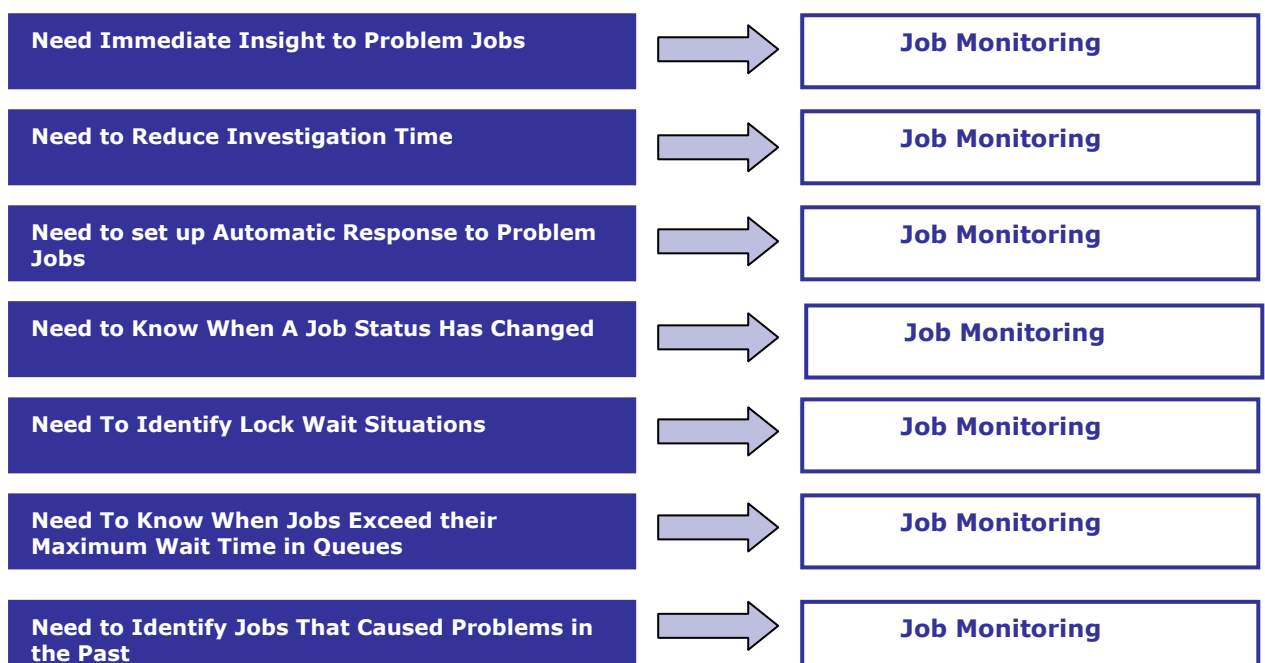
We all make mistakes. When those mistakes, however benign, impact users from carrying on with their work, the impact is immediate and financial. In systems with heavily utilised Job queues, such as QBATCH, it doesn't take a lot to make a mistake that resonates. It may be something as simple as a user submitting a job to a queue in 'held' status and forgetting to release it. Unless it's spotted, that job could be held in the queue indefinitely! Running a simple check that automatically alerts managers to jobs that have been held in a queue beyond a particular time, helps them to proactively resolve issues such as this.

5 Clicks to Problem Resolution

Immediate insight to immediate problems is what makes real-time monitoring a very valuable tool. Equally valuable though is to retrospectively identify issues. This is especially true when teams are stretched for time and have multiple responsibilities. Returning from an important two hour meeting, an IT Manager could be faced with the task of explaining why the system wasn't up to par in the time he was away. He may be immediately aware of a CPU or response time spike and by clicking on that, view the short term history. From there, he can immediately obtain a list of all the jobs running at that time then he simply clicks the questionable job to work with it directly and resolve the issue. Job done!

Evaluate Your Needs

Use the chart below to determine if your job issues could be resolved with a suitable job monitor.



As Company X from our case study shows, unattended job issues can quickly impact an organisation and put a considerable dent in their profits. The more systems there are, the greater the potential for these problems to be occurring across multiple sites on the network. In this case, manual monitoring simply isn't a viable option and problems that take too long to identify run the risk of impacting the user community they serve. When evaluating any potential job monitoring solution, it's important to consider the scope of areas surrounding jobs that can impact system performance. The checklist below offers some key areas to consider when weighing up the functionality of any job monitoring solution:

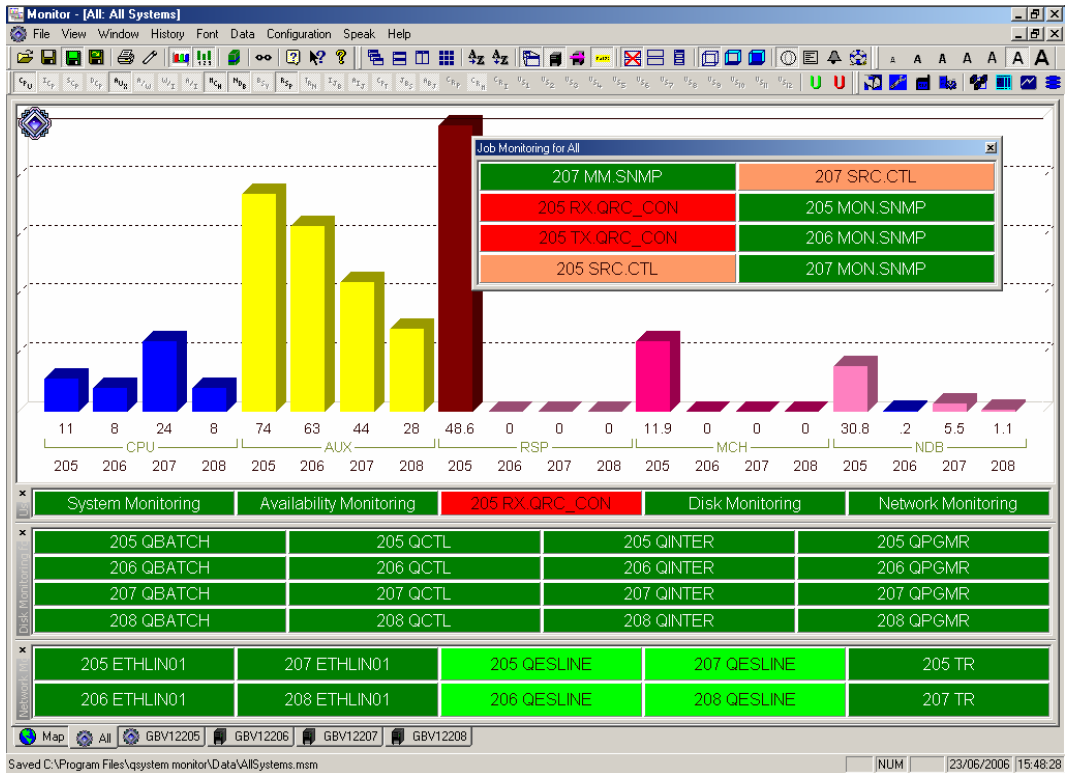
Job Performance and Status Checklist for your System i Network:

Job Performance	Job Status
▪ Response Time	▪ Interactive Job Count
▪ Transactions per Hour	▪ Job Count
▪ CPU per Transactions	▪ Job Queue Active Job Count
▪ CPU Usage	▪ Job Queue Count
▪ Interactive CPU Usage	▪ Job Status
▪ Database CPU Usage	
▪ Faults per Second	
▪ Lock Wait Time	
▪ Temporary Storage per Job	
▪ Thread Count per Job	
▪ Disk I/O	
▪ Job Queue Maximum Wait Time	
▪ Job Queue Average Wait Time	

System Informer

Having a little help on the inside in the form of a job monitor can reap considerable rewards for IT Managers. They will not only be able to resolve issues quickly, but they will also anticipate problems more readily as the monitor identifies specific exception conditions surrounding particular jobs. In the case of retrospective issues, a thorough and unequivocal diagnosis as to the issue can be determined rapidly to ensure steps will be taken to prevent similar issues occurring in the future. For IT managers, the right job monitoring solution could offer the advance warning that saves his company six figures or more annually.

The Screenshot below is an example of the real-time visibility that can be achieved through job monitoring with products available on the market today. Below: The online monitor detects two problem jobs in a defined subsystem and flashes the name of the job in red. The IT Manager has opened up jobs in that subsystem in a separate window and sees there are two jobs (in red) that require his attention.



If your network is struggling with the types of issues mentioned in this paper and would benefit from greater visibility and access to rebel jobs in the network contact the systems management experts, CCSS, to discuss how we can introduce you to a friendly system informer for a life sentence on your network: your new job monitor.

About CCSS

CCSS develops, supports and markets IBM System i performance monitoring and reporting, message management and remote management solutions. An Advanced IBM Business Partner, CCSS develops powerful solutions to support some of the world's most demanding System i environments across many industries including insurance, banking, pharmaceutical and manufacturing. All CCSS solutions are IBM ServerProven.

Existing customers that rely on CCSS's feature-rich solutions include leading organisations such as Volvo, Mattel, Newell-Rubbermaid, The Royal Bank of Scotland, Siemens Medical, RWE npower and Waterstone's. CCSS is headquartered in Gillingham, Kent, UK with key regional headquarters in Raleigh, North Carolina, USA; Bonn, Germany and Makati City, Philippines together with a global agent network spanning Portugal, Brazil, the Netherlands, Switzerland and Sweden.

www.ccsstld.com

