freshservice

The **No-nonsense Guide** to **ITSM**

Practical hints and tips to enhance service management.

James Gander

Visit ***www.freshservice.com*** for the latest in the world of ITSM.

Table of Contents

High call volumes following a change 3

Repetitive calls received for the same issue 9

Ensuring resolution within SLA 13

Breaking information silos within teams 20

Practical tips to enhance service management

If you are relatively new to running an IT operations team or a service desk, you are likely to come across the same things that every other person in a similar role has experienced. From wondering how to cope with the volume and types of calls to trying to understand the changes that need to be made and how. When faced with the pressures of your day to day job, it can be extremely daunting to think beyond the next issue.

I have tried to identify four of the most common issues that managers and team leaders experience at the pointy end of IT, and hopefully have managed to provide some tips and best practices on the optimum way forward.

Problems being experienced

Some of the most common issues experienced by managers and leaders in IT are:

- High call volumes following a change,
- Multiple redundant calls received for the same issue,
- Ensuring resolution within agreed service levels (SLA), and
- Breaking information silos within teams.

My aim here is for you to understand that you are not alone, and be able to pull together your own plans to address these issues so you can feel more comfortable in your day-to-day improvements.

High call volumes following a change

Those on the service desk or on-call, dread those moments when the phone starts ringing off the hook (if that still happens) for no apparent reason. End users are complaining that they are no longer able to access something or run a particular application. So what has gone wrong?

Sometimes this can reach a point when the team doesn't even have time to look into the issue as they are stuck tending to one call after the other for the same issue – leading to increased stress among the team members. How do you handle it?

In some ways, speaking like a true consultant, it depends on the size of your organization and the maturity of your work processes. If you are a team of 4 people in the same room, then identifying the potential cause of the issue comes down to the team turning to each other to ask if anyone has done something. If everyone present doesn't have anything to add to that, then it is either a failure, or the person who does have knowledge about the issue isn't in yet.

However, if you are part of a larger team across multiple sites with mature processes, then you might check the change calendar, identify if any of the implemented changes may have had this effect, and contact the relevant technician. Of course, there are many options in between – including unapproved changes which even the most mature of organizations have to contend with occasionally.

At this point, however, you primarily have the following tasks to carry out:

- Handle the current situation
- Communicate to the users
- Plan a fix

Handling the current situation is relatively simple. Unless it is email that is down, send out an email notification to those affected. I suggest email because in most organizations, it is the most widely accessible form of broadcasting information to IT users. The email could be to particular departments, sites, teams, or everyone. Tell them that you are aware of the issue, that the appropriate teams are investigating, and set expectations on when you will provide an update. Every 30 minutes is good to start with, but if it looks like it will go on for hours, make it hourly or whenever it's possible from your end. This may not be the whole truth, but it won't be far off. Users who know that you are dealing with the issue are less likely to harangue you with calls, and more likely to accept the outage. Of course, you won't get everyone but **if you can cut phone calls by 75-80%, it does a lot for reducing stress levels.**

If email is down, this can be trickier. Experience tells me that it is always worth having an alternative method of communicating with users. One of the simplest methods is to have a list of key people in each site/ department/ team and their phone number. If the SD manager/ SDM/ IT manager has an SMS group on their mobile phone, they can easily send out updates requesting to communicate about the outage with their teams and those in their vicinity. However, there are a multitude of options, and some of you might prefer corporate approaches. Maybe you have an intranet that everyone actually uses, checks regularly and considers as the first port of call for any updates. Maybe you have a great service management portal, and all your users look to the portal, by default, for communication updates, or you use a third-party tool for communication. Whatever it is, **make sure you don't rely on just one method to broadcast information to users.**

Planning the fix will depend on many different things. It can be as simple as having the technician who implemented the change look at it and say, “Whoops, my mistake. I know what I did there. Give me two minutes,” and the change request is updated accordingly and everyone is happy. Alternatively, no one may have a clue as to what is happening. The change tested successfully, the implementation went according to plan, so ideally, nothing should be causing an issue.

If you are at this point, it is time to raise the question, “Can we rollback the change?” If so, plan for it only if the fix cannot be implemented within a reasonable time-frame and it is safe to do so. If not, then you will need to start to document something like the following:

- What are the symptoms?
- What is different to normal?
- Can the issue be reproduced? If so, how?
- Is everyone experiencing the issue? Is it the same issue?

Start to narrow it down. There will probably be a multitude of options from the technical teams as to what the issue is, or might be. There will also be strong feelings about why something cannot be the cause. Mapping them all out on a whiteboard or screen where everyone can see them is a good way to drill down the issue. This exercise needs to be completely unbiased and non-judgmental.

We should always work in a blame-free environment that encourages people to safely try different approaches to work, as that is what helps drive improvements, but reality tells us that some organizations don't work like that. If you have an option and the consensus is that it can't be the issue, ask why until all the options have been disproven. Then, move on to the next.

Communication between teams and the users is vital. For users, you should set expectations that you will provide them with hourly updates, and make sure that happens. Make it somebody's responsibility to do that, and somebody else's to check that it happens. You only need to tell users what the issue is, and whether you have fixed it yet. They don't need details. Try and use the same format of email each time so that users don't have to read paragraphs to get the information. If possible, try tables with single sentence updates.

For your teams, communication is imperative. The service desk, dealing groups and service delivery manager/ service owner need to know who is doing what, when, and why. Those working on fixes need to know who to update and when. This may be the major incident manager, the service desk or the service owner. It doesn't really matter, as each organization is different and there is no right answer. However, **knowing who is doing what and when saves time, stress, reworking and looking stupid in front of users.**

Once you have identified the fix, don't jump into it. Likewise, don't allow others to be trying things while you investigate. If that happens, you will have no idea what the cause was and what the fix was. You now need to consider whether your fix will impact anything else. Do you need an outage of another part of the system? When can that be done? All this will need to be communicated to your users again.

But how do you stop these types of issues? In short, you can't completely stop them. There will always be the unknown. However, there are certain things that you can do to mitigate the chance of it happening. A change management process of some description will help. It doesn't need to be a heavy process. It just needs to fit your organization. Maybe, every change to do with servers and networks has to be logged. Maybe, they all need to be reviewed by the team leader. Maybe, once they have been done 5 times successfully, they are classed as pre-approved and they only need to be recorded. Maybe, you have a tool that will allow them to be automatically tested. Or maybe, that can be integrated and the whole change is automated. **The less human interaction, the lower the chances of something going wrong.** So, if an application is updated, the test team (if you have one) could look to automate an end-to-end test plan for the application. This would ensure that every component used to deliver the service works with the update. This could be used by the application team even before they raise the change, and if the results can be easily shared, the change becomes pre-approved, assuming the tests were successful. If that is too much for your organization, maybe a test plan could be written, reviewed, and signed off by all teams (apps, infrastructure, DBA, networks, the business teams, etc.) that can be centrally stored, and followed by everyone involved in making any change that affects services. Then, if something changes to a service, the centrally managed test plan is automatically updated, and everyone knows they are covered (hopefully).

Repetitive calls received for the same issue

We all know of those endless streams of calls where, over the course of a week or month, you receive multiple calls for the same issue. Or worse, maybe you aren't aware of the fact that you are handling many calls for the same issue. What do you do?

Well, if you aren't aware, then you could look at the way you are categorizing calls. I'm a big fan of keeping the number of categories short, but using sub-categories to help clarification. For example you might have application, email, calendar, or desktop, OS, Win10, or communication, VC, and cabling. Whatever you have, needs to work for you and your setup, but it also has to be easy for the guys at the pointy end of support to be able to understand so they can help quickly. It also should be replicated when you resolve a call. Occasionally, what you thought the issue to be, when logging it, might be different to what the issue was. And there is no point in reporting on what you thought the issue was, but what it actually was. That said, if you have people constantly entering the incorrect categorization initially, there may be a need for training because either they don't understand the issue, or the issue wasn't communicated properly to them.

Once you gather the correct information, you need to report on it. This will identify trends, some of which are acceptable and some that will be a cause for review. You may see that 30% of your calls were to do with a particular application, but not about any particular issue. Do you need to investigate further? Maybe not. You may see that 30% are password resets. Should you do something about that? Probably, yes. It may be worth investigating a self-service platform that enables users to reset their own passwords. However, funding to implement self-service will need additional work.

What does it cost for the service desk to handle all of those password resets? How long is a user “off-line” while waiting? What is the number of out-of-hours calls for password resets? What would it cost to fully implement a self-service solution? Not just buy it, but implement, integrate, communicate and train users and support staff. There are multiple hidden costs that can cause future trust issues if they aren’t identified and understood initially when requesting funding. How long will it take? Does the CIO expect a reduction in password resets within 1 month of sign-off or 6 months? All of this need to be considered before beginning process for funding.

Your reports to understand what is happening will probably need to be looked into over a period of 1-3 months before you identify any meaningful trends. Sometimes, this can take longer. It is highly likely that you will have anomalies every so often, where a peak in calls will skew your metrics. Maybe you had just released new functionality in an application and for 2 days your calls went up. Is that a cause for concern? In the long term, maybe not, but it may identify a failing in the project delivery. Was there enough training or communication? Should there be more super-users walking the floor in the future? Or was it really just one of those things that everyone knew would happen because that team needs special treatment?

Once you have your metrics and you have identified a few pain points, you will need to prioritize the work and understand the actions required. Are the actionable tasks something that you and your team can do on your own, or do they need support or involvement from others? If you can handle it internally, then work out when it should be done, and allocate time accordingly. If you feel that you need to address the issue by the end of the month, get it done.

Do you need to take somebody off the desk for one day a week to achieve it? Then do it. If you need involvement from another team, you may need to sell the idea to them. Does the service owner or product owner need to add the idea to their improvement plan or backlog? Does the infrastructure team need to allocate someone to work with you on the resolution? Do you "just" need documentation from somebody? That may require a couple of hours, twice a week with somebody to write, review, categorize or tag, store and communicate.

This can be seen as problem management and in some instances, it is. You could have the problem manager co-ordinate the work that needs to be done by multiple teams. Or maybe you have service owners and they will do this in order to reduce call volumes for their services. However you do it, where there is cross-team collaboration required, you need buy-in. Maybe this is a great opportunity to implement virtual teams, who work together to drive improvements in small sprints. If that works well, why not keep that team as a team that supports the particular service? You are taking your first steps towards a DevOps set-up. The team could work through a list of improvements that are broken into smaller manageable chunks which are then prioritized by an owner.

Ensuring resolution within SLA

SLAs. Service Level Agreements. The bane of every support person's life. Are they needed, and if so, are they defined and agreed upon in a sensible manner? It is highly probable that either somebody within the wider business made a decision that "priority X" calls would be responded to within "Y minutes" and resolved within "Z minutes." It may also have been agreed that systems won't be unavailable for more than "ABC minutes" a month or year. How realistic are these numbers? If the wider business didn't dictate the service levels, then maybe someone in IT said that they should be "this" because that is easily achievable and the users complain about it taking too long to get anything done.

Let's assume that within your organization, SLAs have been set as required. We are talking about internal service providers (IT departments) here, but external service providers are not that different; they just charge "real" money rather than internal costs. How can you be sure that you can provide solutions within SLA? Well firstly, let's remind ourselves that SLA stands for Service Level Agreement, so it is an agreed level of service which will be provided. Agreed. So what should ideally happen for a service is that the service provider sits with the service customer and agrees on the service level. The customer will want X and the provider will want to provide Z. Hopefully, they can agree on Y. This negotiation for a brand new service in a traditional IT environment will go something like:

> ***Customer:*** I'd like the service to be available 24*7 and all calls to be responded to within 10 minutes, and resolved within 20 minutes.

Provider: Okay. To do that, we need to design the whole solution to be resilient with no single points of failure and hot standby infrastructure in a secondary data centre. That will cost three times the current budget. To respond to and resolve all calls, no matter what they are, within said timeframes will require investment in these tools, training and additional people at a cost of x. Can you sign the CAPEX off, and ensure that increased OPEX costs are agreed for the next 10 years please?

Customer: There's no more money.

Provider: Then based on current budget I can provide…

There will be a bit more negotiation and some shaving here and there, but eventually signed off at a hopefully realistic agreed level of service for the available budget.

If a DevOps setup is in place, then some areas will change in the discussion, some might be able to deliver more and quicker for lesser costs, but fundamentally, if the investment isn't in place, there is only so much that you can do.

Now, we all know that in the real world, the above is unlikely to have taken place. So people will expect you to occasionally pull one out of the bag. And you will because IT always does, but it leads to expectations being greater.

Firstly, let's accept that it just won't be possible to always deliver within SLA. Even if the SLAs are realistic and have been designed to be achievable, things happen and get in the way, stopping you from delivering what is wanted all the time.

If you are an internal service provider, that may not be a big deal, unless you are constantly failing to deliver within the SLA. Then, conversations around outsourcing may raise their heads. That's a separate conversation. If you are an external service provider, I would like to think that contracts are not brought up whenever you are perceived to have failed to deliver. I would like to think that you are having grown up conversations with the customer explaining what happened, why and what is being done to mitigate the chances of this happening again. Those same conversations should be happening for internal service providers as well. You're all in the same boat.

So how do you really provide solutions within SLA?

When managing incidents, you need to have visibility to be able to remove the chances of incidents, or to be able to restore service within SLAs. The two most obvious methods of doing this are monitoring and knowledge.

Monitoring

If you are effectively monitoring your services, you should be able to identify and address most of the issues before the users are aware of them. Alerted about low disk space on your storage? You can do something about that BEFORE it becomes an issue. Or if it has suddenly become an incident that is causing issues, you are ahead of the curve and handling it, with the purpose of resolving within those SLAs. Of course, there will be sudden incidents that no amount of monitoring can pick up and alert you in time. Network devices can just pop, and so can servers. Design correctly in the first place and that shouldn't cause any undue stress. However, live in the real world with limited budget and resources, and it will. We will cover how to handle that soon.

Knowledge is vital

If your support teams, whether service desk, infrastructure, application or any other team, don't have easy access to knowledge base articles that show how to resolve issues of certain types, then no amount of technology is going to help. Transferring knowledge to people's minds is also not going to work. How you address the issue of the people who don't want to or won't share their knowledge is a separate topic altogether.

Requests

Most service desks have to handle more requests than incidents. That's good news, on the whole. However, if they are handling these requests through paperwork and manual processes, it will delay service delivery and potentially violate SLAs.

The first step to addressing this is to migrate all requests into a request catalog. Not a service catalog, which details the services IT provides and the SLAs but the request catalog which details the items that you can request. Very different things.

With the request catalog, you should be able to make it a clickable process, where the user needs to do as little as possible. If approval is required, set that up in the process so that the user does not need to send a separate form to a manager in a different building to get it signed, and then have it sent to the service desk that has no idea whether the signature is the right one or not and will process it, hoping that they are not breaking some policy. You could build communication into the workflow of a request through the catalog, so that the user gets informed where their request is, its reference number, when to expect the next update, etc.

If the user is requesting software, maybe you could look at automated deployment of the standard applications, so that IT doesn't need to manually work on the install. Maybe, for new users, you could have HR approval as part of the process and create an AD account with all the right access permissions so everything required for a new employee is ready even before they turn up. This won't magically happen overnight, but understanding your constraints and bottlenecks in the request process, and how long each step takes will enable you to improve the flow of work and deliver more easily against SLAs.

Going back to a point made earlier about budget and resource constraints, this doesn't help provide the level of service that you customers might be after. I mentioned about designing the service correctly in the first place, so that a component failure won't impact the services. Well, the realism fairy has just flown into the room. IT may want resilient hardware but the customer may not be willing to pay for it and yet still want the same SLA. How do you manage that when something fails? Not easily. However, communicating during an incident helps. Don't lay the blame at the customer's door. You can't say, in public, that you wanted to have resilience but they didn't want to pay for it. Well, you can, but that won't help anybody.

What you can do is throw as much resource at the resolution, and make sure you have knowledge base articles that cover as many scenarios as possible. Make sure the service design is documented so people understand how it works.

Do everything that you realistically can to restore service within SLA. Communicate all updates as regularly as possible. Then, when service is restored, produce a report that contains no emotion, and details what needs to happen to reduce the chances of this happening again. If resilient components will help, mention it, but without dragging up all the other times you mentioned it. Maybe an outage isn't as painful as the business expected and therefore, they don't want to spend more on the service design. Maybe it was and they can find the budget to do want needs to be done now. Open and honest, non-emotional communication will help everyone get a clear understanding of what needs to happen to move things forward so that either SLAs can be met, or revised to be more acceptable to all parties.

Breaking information silos within teams

This is one of the trickiest things out there. It's about the people in the teams and the culture of the organization, not just tools. Asking people to share knowledge or information better within teams, in order to break down the silos that have grown over the years, is hard and cannot be done by one or two individuals. This needs to be driven from the top. Yes, a team can do this themselves, but as soon as they meet the immovable force that is "the hero," it will stop. **IT leaders need to make it clear that those who keep information to themselves help no one and will not be accepted.** They may feel that they are the hero because they have to be called up on to resolve issues at all hours of the day and night, and even when on leave. They enjoy the need that people have for them, and then they love to complain about never being allowed time off. It's always someone else's fault. But they won't share knowledge because others can't be trusted. The only way to stop that is to make it clear that the new heroes are those who share and collaborate. If people don't like the need to share knowledge, then maybe they aren't the right people for your organization. Do they fit your culture? They might have done in the past, but if you are wondering how to break this mindset, then it needs to be acknowledged that they are no longer the right people. It sounds like your culture has changed, but some people haven't.

Are these "heroes" really a problem? Well, how often does a service restoration get delayed while you wait for them to be available? If they make themselves available immediately, what doesn't get done? Are they holding up projects while fixing issues? As with anything that we do in business, we have to identify, understand and remove bottlenecks.

How do you do that? Start by explaining to the teams what you are doing and why. Get the CIO to take part in this and visibly support you. You could experience pushback from senior managers who like “the hero” because they always fix the manager’s issue quickly. Not sharing knowledge or information is a risk to IT as well as the business. Identify why people won’t share knowledge. Is it a trust thing? Is it purely because they don’t have time? Is it because they don’t realize that they are doing this? Make it clear that this is no longer a negotiable aspect of their role. Will they leave? Possibly. Will that hurt IT? Again, possibly. However, experience has shown that IT may hit a few bumps in the first few weeks of the hero not being available, but everyone steps up and works it out. No one, unfortunately, is indispensable.

If possible, look at creating scripts, so that the ability to perform certain tasks doesn’t require permissions for individuals, so it becomes less of a security risk and removes some of the trust aspect. Maybe you could get the “heroes” to produce carefully documented knowledge base articles – if this, then that kind of documentation with screenshots. If they won’t, or can’t do that, get somebody to sit with them and, as part of their learning exercise, have them produce documentation that is signed off by the “hero.” If only certain people in appropriate teams are trained and provided with the information, and only they are allowed to do the fix with supervision initially, then alone, this reduces another element of the lack of trust. I was told recently about the 1:3 3:1 principle, where 1 person should be capable of doing three tasks, and three people should be able to do one task. This adds redundancy to everything being done and reduces risk.

Finally, to share knowledge and increase collaboration, you could consider creating virtual teams, so that rather than only sys admins being allowed to do certain tasks, virtual teams can be created for each service or product. The team could be made up of sys admins, service desk, app support, etc. and when a call for a particular service comes in, it is assigned to the right virtual team. They could then share knowledge amongst themselves and become less self-reliant.

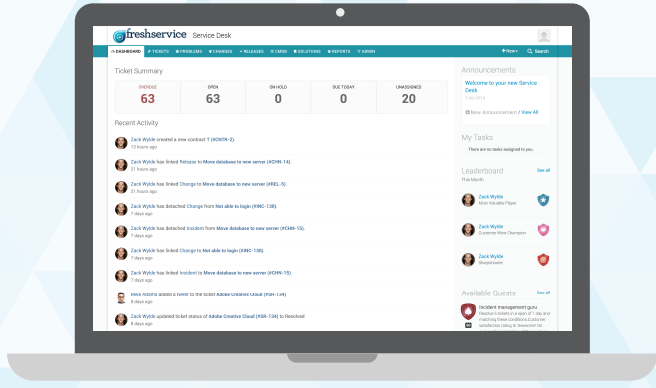
While there are plenty of other issues that need to be covered, these are the ones that commonly hinder operations across organizations. Hopefully, this white paper helps you break down complexities in your service desk to an extent.

About Freshservice

Freshservice is a cloud-based IT service desk and IT service management (ITSM) solution that is quick to setup and easy to use and manage.

Freshservice leverages ITIL best practices to enable IT organizations to focus on what's most important – exceptional service delivery and customer satisfaction. With its powerfully simple UI, Freshservice can be easily configured to support your unique business requirements and integrated with other critical business and IT systems.

Native integrations are provided "out-of-the-box" with many of the most popular cloud services such as Google Apps, Dropbox, AWS, and Bomgar to speed up deployment and reach. Freshservice is built on the proven Freshdesk platform, whose flagship customer service offering supports more than 100,000 customers worldwide, including Honda, 3M, Macmillan, Bridgestone, and Unicef.



freshservice