

;login:

THE MAGAZINE OF USENIX & SAGE

July 2001 • Volume 26 • Number 4

inside:

WHAT ARE YOUR INTENTIONS?

by Brent Chapman

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

Why are we using a given package or version for a particular service?

Why Postfix? (rather than Sendmail or QMail or whatever)

Why Oracle? (rather than MySQL or Postgres or whatever)

Why BINDv8? (rather than BINDv9)

Why RedHat? (rather than Debian or SuSE)

Why are we using a particular vendor or service provider?

Why Dell? (rather than Compaq or Micron or whoever)

Why Sun? (rather than HP or SGI or whoever)

Why UUNET? (rather than XO or PSINet or whoever)

Why Exodus? (rather than Level3 or Digital Island or whoever)

Why are certain services provided by particular systems, rather than some other system?

Why are certain services grouped onto particular systems?

Why are other services given their own systems?

Why are our standard systems what they are?

- Why this config for an engineering desktop?
 - Why this config for an engineering laptop?
 - Why this config for a non-engineering desktop?
 - Why this config for a non-engineering laptop?
- Why is our IP address plan set up the way it is?
 - Why do we allocate a /24 to each office?
 - Why do we skip every other /24 in the allocation?
 - Why do we allocate all the point-to-point /30s as we do?

Competent IT professionals usually have good reasons for the things they do. They can usually tell you those reasons, if you ask. Months or years later, however, it can be difficult to obtain this information; the professionals in question may have worked on so many things since then that they no longer remember, or they might not even be with the organization anymore. A much better approach is to create a short explanation (maybe just a paragraph or two) of why a system is the way it is at the time the system is configured.

These reasons are often discussed and debated among staff while the system is being designed and deployed; all you need to do is capture the conclusions of those discussions and debates. At a small shop, an archive of your internal IT email discussion list might be sufficient, if you've got a good search mechanism. Or you might want to go one small step further and establish a special archived email alias just for these "wisdom" summaries; the trick will be to remember to create such summaries and send them to the alias for recording.

Growth plans are a related issue. Competent IT professionals usually include avenues for efficient future growth in their designs, but those avenues often aren't obvious later, especially if the person looking for them wasn't a participant in the original design process. These growth plans and thoughts should be documented when the system is designed and deployed, so that they can be used when needed in the future.

Assumptions about the environment and its future are another thing that should be documented when a system is designed and deployed. Documenting these assumptions will help later IT staff determine when a design is no longer suitable for the current (ever changing) situation, and should be revised or replaced. For example, if you design an IP address plan to accommodate a dozen large field offices, total, over the next five years, but your company has shifted plans and is instead now opening a dozen small offices per month, you probably need to rethink your address plan.

Assumptions can also take the form of constraints, which might not apply in the future. If you don't document them at design and deployment time, however, it will be more difficult to take advantage of later changes that ease those constraints. For example, you've probably heard the joke about the mother teaching her child their family's traditional method of preparing a roast, which starts with cutting the roast in half. The child asks "Why?" and the mother says, "Well, that's the way we've always done it; that's the way my mother taught me to do it." The child, being a naturally curious sort, goes and asks Grandma "why cut the roast in half?" and gets the same answer. The child recurses through ancestresses until eventually reaching one who answers, "Because it wouldn't fit in the pan that I had when I was a young bride." That's a perfect example of a system carried forward far longer than necessary, because nobody realized that the relevant constraint (the size of the pan) no longer applied.

To get a good idea of what your "why" documentation should cover, imagine the questions that a new hire would ask if they were made responsible for that system or service.

Assumptions about the environment and its future are another thing that should be documented when a system is designed and deployed.

As a consultant, asking “why?” is often the most important service that I provide to my clients.

Assume that this person is a competent system administrator, and knows how to find and use vendor documentation for the system or service; they’d still want to know things like who the primary users are, why this particular platform and/or software package was chosen to provide the service, what the growth plans are, what the unusual or subtle aspects of this configuration are, and so forth.

As a consultant, asking “why?” is often the most important service that I provide to my clients. The act of explaining their concerns or goals to an outsider (me) often clarifies those concerns and goals. This, in turn, often makes a range of solutions clear as well, and the task becomes one of evaluating and choosing between those solutions, then implementing the chosen solution. There’s no magic to hiring a high-priced consultant for this type of exercise (though I certainly don’t mind the business!); you can apply this same method very well yourself, if you just take the time to do so.

So... what are your intentions?