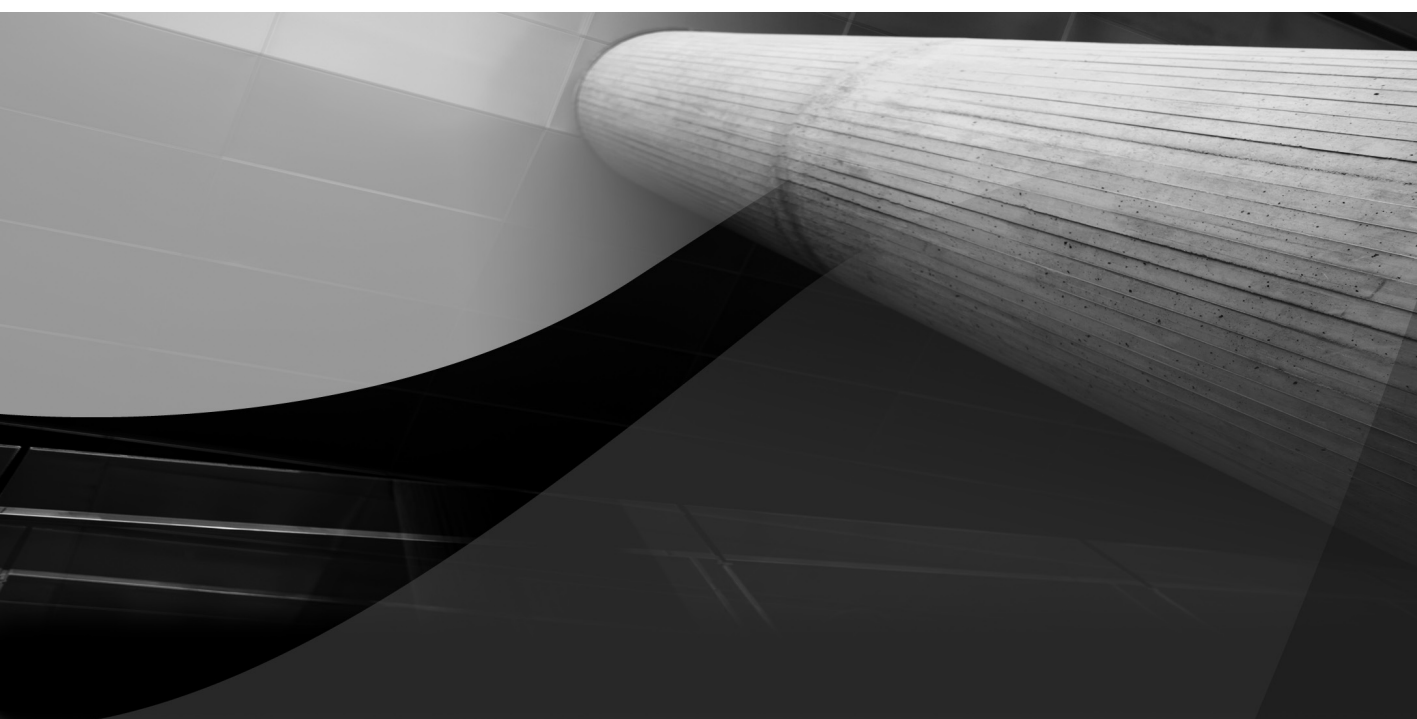


PART II

Preparing the Enterprise for IAM



CHAPTER 3

Planning an IAM Project

A fine invention is nothing more than a fine deviation from, or enlargement on a fine model.
—Edward Bulwer-Lytton



You don't go on vacation, change apartments, buy a house, go to Grandma's for Christmas, or do anything else in life, without a plan. Why are we doing this, what are we doing exactly, what do we need to bring with us, what are we going to accomplish, and when do we get home?

You may or may not be planning an IAM project in anticipation of a second pass at management for approval. This depends entirely on the organization. If you've had enough ongoing issues, and management is already on board with the idea of IAM, you might already be empowered to put a working plan together. But I know what it's like to help a customer do a first pass with management, followed by a plan, followed by a second pass at management in order to validate not only the need for, and business value of, an IAM framework, but also the ability to deliver a solution. In other words, before they give you the green light, they want to make sure you're not going to crash, so they want to see that plan. We'll cover in Chapter 5 the ins and outs of putting together a more business-based presentation to management, in order to make a stronger business case, which will in turn secure budget and resources.

Resources, Both Digital and Human

Certainly putting together an IAM framework is a considerable investment in time, money, and manpower, and requires a plan. How do you go about such a daunting task? From start to finish, in very basic summary (which you'll break out in detail when you make your business case to management), let's outline what you need to put together.

The Processes

This is all about process to begin with, so start there. You must secure and augment the business. You must secure assets (the resources that actually run your business), enable users, get those users in and out of your system safely, and ultimately report on all activity. You must define those processes, including the setting, the actors, the actions, the reactions, and the end results. And of course, you want to *automate* as many of these processes as possible. Keep thinking *automation*. And by the way, when you get to the point of looking at buying or building, that's when you expand those process plans into use cases that reflect specific real-world situations.

- **Creating accounts** You can't enable users until you give them a place to live. How will they come into the system? Bulk load? HR event? Self-registration? Will an administrator key them in? It will likely be all of the above, depending on the type of user, and you will have multiple types. Define the processes first. We'll get to the actual landing in a moment.
- **Provisioning** How will users get access to their resources? Automatic grants based on attributes or roles? Direct assignment by managers? Resource request by self or manager? A combination of these? What approvals will be needed for requests? And what information must be communicated to the target systems so that they recognize the new users when they come calling?

- **Change management** When a user is promoted, demoted, transferred, or terminated, what needs to happen? Remember, termination is simply another state. A darn important one, but it's a change, just like a transfer, except that the terminated user tends to lose a lot more access than a transferred user.
- **Authentication and authorization** What information must a user provide in order to access resources, and what is the process for that authorization? This *must* be automated, obviously. They are two different processes, but they must happen one after the other. I log in according to requirements (password, token, pin), and then the system must authorize me right away so I can get to the good stuff. As part of that, what information must be communicated to the target system, and by what method, so that a properly authorized user can gain entry?

All these processes we just discussed will generate a lot of logging. What data is captured, *how* is that data captured, where does it go, how is it recalled, and in what form?

Everything from here on is only meant to feed those processes. This is all about getting users in and out of the business resources in a secure manner. Define those processes, and the pieces should start to make themselves obvious—you would hope. What's next?

The Requirements

You say you need to be able to get to the store, buy a lot of groceries, and get it all home. That's your process. The first requirement is transportation. So start thinking about your options.

- To fulfill the process requirements, there are supporting definitions to create. These are the objects (and yes, a user is an object) to be defined which will be players in those processes. What does a user profile look like? How about group structures? Role definitions? How will these work together? Will groups serve as roles, or vice versa? Will roles or groups be assigned via user attributes? Before you can assign privileges to a user, you must define what a user looks like, and what he needs to have in his pocket before he can receive those privileges.
- Figure out the extras you need, those gaps you need to fill, in addition to what you already have. What's not working? What's not there? What are the deltas between current and desired functionality? What is on everybody's wish list? Automation? The ability to remediate? The ability to quickly terminate? Providing access to external users? This is not to say that everything you want, or even everything you think you need, will result from this process. But there are needs, and there are *critical* needs. Focus on the immediate needs of the various parties, and how those needs might intersect, as well as how the solutions to those intersecting needs might complement each other. And while you're at it, find out what your auditors can do for you. They can be a font of information on your deltas. "During the last audit, you failed to provide these reports."
- Don't forget the things you haven't even thought of yet: new audit/compliance requirements, new resources, new user groups, new technologies, and anything else that will require agility and openness on your part down the road.

The Pieces

Now that you've figured out that you need transportation to get to and from the store, you extrapolate that requirement to mean that you need a car. So now you think in terms of the pieces you need to satisfy the requirements for your processes.

- You *will* modify your perceptions of the technologies you need as you progress and investigate what's available to buy and/or build. But you should have some notion of at the least the technical functionalities you'll need to have in place to make this work. Don't even think right away about building versus buying, but simply about actual functions.
- Inventory what you already have. What data stores are in place, and how is the data used? Which applications are operating; how are they secured and provisioned; what communications do they have with other apps; what are their data stores? What are your existing resources, and how can they help you? These can be human as well as digital assets, of course. Do you have a workflow engine? How about help desk software and personnel? You've already inventoried processes. Now determine how many of them are manual. Remember, automation.
- Build a list of what else you need—what in your identity and security infrastructure isn't working, and what the deltas are. What are your auditing requirements? What was needed for the last audit? What will the auditors want this time around?

The People

To help gather this information, to validate the requirements, to help build the case, and to provide the resources, you will need to inventory the people available to you, or at least the people you'd like to have available. There's no guarantee that when you identify a stakeholder, you will get that person on your side.

- Get management on board with the notion of providing a framework. Show them what you've put together. Make sure the people with the wallets will even let you think about possibly spending some of their money to upgrade. If the answer is no, you might still want to spend some free cycles assembling elements for a business case that might get a better hearing when the business environment improves or, heaven forbid, you suffer a breach. If the answer is "maybe" or "yes," then go ahead.
- Identify stakeholders and champions. Figure out who's on your side. You will need them. I repeat: You will need them. Sell them on the value. Remember, you may not get everybody on your side all at once, so occasionally it helps to get one or more stakeholders to help you get even more of them on board with the plan. One big suggestion: When you're trying to entice somebody to integrate their resources with the framework, be careful not to insult what they already have. Unless their homegrown, silo-based security has caused a breach, you can't beat it up. Simply sell the value of participating in the overall security umbrella. If you call their baby ugly, they in turn will set very high expectations for you.

- Form committees to set standards and requirements. If the company will let you outline all the needs yourself, based on your cursory investigations, you will still want input. You won't think of everything yourself. And even if you could, somebody will invent something else. If the scoping and building are limited to a small group, then that group is liable. This is the opportunity for people to make their voices heard. Get a comprehensive view from the crowd, and also (sad to say) limit your liability. In other words, more voices will lend obvious weight to your arguments.

The Resources

These aren't solely your digital resources. It includes the human capital you need to get this project off the ground.

- Explore buy vs. build compared to buy-and-build. Some stuff you'll code yourself. A lot of it you probably won't. Start searching, and match up your desires with what's available. And always remember, internal staff will often want to do it themselves, for job security reasons, and they will almost always *underestimate* the time it will take. You will be involving them, for a variety of reasons that we'll cover later, but you should remain in charge.
- Decide on whether to get help. Whether you buy or build, or buy *and* build, can you scope, acquire, build, manage, and deploy on your own? Or will you want outside experts? Perhaps you've already got relationships with consultants who can help. Remember, these people are the opposite of your internal staff; they will almost always *overestimate* the time it will take.
- Determine the levels of time and investment. How many resources will this take? You will surely be pulling people off their regular duties, and they will be drafting additional equipment. It will cost money and time, which is actually money *twice*. Don't discount the amount of effort you and your staff will be devoting to it. It's not just the software and consultants; it's you. Your time is valuable. A very basic deployment plan will help.
- Engage vendors and integrators. Start dragging in the other organizations who may be providing software and/or labor. Vet them thoroughly, their capabilities, resources, expertise, estimates, references. We'll be going over this, believe me.

The Design

You can't just pick up the pieces you need. You must fit them together, so that collectively they serve your process requirements.

- Knowing the software and hardware and labor requirements allows you to compose a more detailed design: what's prebuilt, what needs to be built, what's kept, what's tossed, and who's providing what.
- Make the business case to management for funding. Once you've got it all tallied up, you go to the bosses and say, here's what it will take.
- Design review. Let everybody beat it up. And this they will do. Wear your thick skin to the meetings. Sure, it's the same people who helped you come up with the requirements. Doesn't matter. Things change.

- Determine internal resource requirements. Now that you've vetted the outside folks, return to the inside folks. You will need to recalibrate your internal resource requirements after engaging vendors and consultants.
- Implementation plan. What happens in what order over how long? What are the timetables? How do you set milestones and deliverables? Who is responsible for what pieces? In what stages or phases will the deliverables arrive? You will not be doing everything in parallel, and even if you plan to do this in steps, which you should, all the things that go wrong in the first step will delay the second step. Just expect it.
- Final design. Everybody's had their second and third say. Roll it all into the plan.

The Roll-Out

It's not enough to figure out what you need in terms of process and pieces and then create a design. You need to know in practical terms how you will implement.

- **Acquire and/or build** Start matching up software, processes, and use cases. This phase includes construction, unit testing, and enforcing timetables and deliverables.
- **Phased deployment and testing** Actual rollout of pieces to departments. More in-depth testing.
- **Training and hand-offs** These go hand in hand with the previous step. Train the admins on how to manage the new pieces and processes. Train the end users on what to expect.

Remembering the Goal

You won't be doing this alone, right? You're going to solicit a lot of input, as well as feedback to your *own* input. It's easy to get distracted once you've engaged all the special interests within the company, because while you might have a purpose in mind, the herd may take you in different directions. Remember to lead, and to periodically remind people of the goal(s) of your project:

- The final IAM system must serve the entire organization.
- It will take more work up front, but ultimately will save on labor.
- It must *reflect* and *augment* the business. If your company generates its profits by making widgets, it should help you make those widgets. If you sell insurance, it should help you sell insurance. IAM's not widget software? You're right. But it should allow the line-of-business people to concentrate on the widget business by taking identity and access management off their shoulders. IAM should model the organization and its job titles. It should let people in and keep people out, as dictated by business requirements. It's all about the business you're in. This is just a helpmate. It has no reason to exist except to help the business. The variations should only come when you see the opportunity to *improve* your processes. This should be apparent early and often; otherwise, why are you doing this insane thing?
- Any suggestions made by your committees should be to improve processes or make up for shortcomings in the way you do things now. Don't allow the little wish lists to take you off target.

Periodically review your high-level goals, and use them as a barometer of any recommendations or requests. Do the recommendations help the business? Do they reflect the business? Do they get you closer to helping the entire organization? Do they meet integration, reporting, and audit standards? Do they fall within the domain of what you set out to do, or are they part of some provincial need that somebody tried to tack on so they could ride your coattails? Nothing attracts followers like a project with a budget. Don't be somebody else's meal ticket. Stick to the plan.

Getting Ready to Break Things

When I was in the process of publishing my first book, I was reminded by my editor more than once, with a phrase that made my wife cringe, "Don't be afraid to give up your babies." In other words, you fall in love with something you've created—a paragraph, a phrase, a piece of software, a piece of ugly furniture you kept from a long-ago yard sale—and you don't want to give it up, even when it doesn't fit any more. But if it's in the way, if it's part of the problem, or if it simply isn't part of the solution, consider it a candidate for setting aside.

Your manual processes are working? Wonderful. But they're manual. E-mails from the boss get the approvers to move on requests? Great, as long as he remembers to send them, and he's not on vacation.

That's not to say everything's got to go. Let's say you're provisioning to customer relationship management (CRM) via some help desk app, like Remedy or Magic (which usually means opening a ticket to push requests that are ultimately fulfilled manually). If it's working, maybe you don't want to mess with it, at least for now. So when you put in your new provisioning system, perhaps you can make that help desk app one of the target systems which proxies for CRM. You may or may not change how you provision to CRM later.

Or you might say, why have the help desk app serve as a middleman? If it's going to slow up provisioning to CRM, why not put the CRM package on the framework? Remember, that's the point, to build a framework that your apps simply hitch a ride on.

One point you may wish to make to any potential vendors or partners is your desire to preserve as much of your existing investment as possible:

- Don't spend money where you don't need to.
- Don't make the transition any more difficult than it needs to be.
- If it ain't broke, don't fix it (although it might need tuning).
- If changing something will drastically interrupt service, we'll attack it later, or just differently.
- It might not be the best, but it functions for now, as long as it doesn't get in the way.
- It won't get in the way of the new stuff that we absolutely do need to build.

Whenever you take on a big project, you should expect to hear from management, why do you want that? And this is a good question. Sometimes you're so used to doing things a certain way out of necessity, you might come to find you don't need to do that any more because the new system will bring you improved processes. Here's an example.

At a huge financial services customer, one of the requirements was to perform data imports from a weirdly formatted file. Yeah, we could do that, but it was a pain, and early on we were smart enough to ask, *why do you want this?* And the reason was that their old request system

required incremental data dumps from the old HR system to keep users in sync with entitlements, and that was simply the way they'd been doing it.

We said, that's dandy, but guess what? We have a connector for your HR system; we can query the data directly and bypass the whole dump–message–upload process. On top of that, once we have the users and the entitlements in hand, we manage the synchronization and periodically reconcile users with entitlements, ensuring least privilege. So sure, we can do that upload thing, but you won't need it any more.

I spent literally years pitching, strategizing, implementing, and helping to support a multimillion-dollar IAM project at a global manufacturer. My company was forced, for political reasons, to partner with an integrator who had impressed the client. The integrator brought in a particular brand of a barely known third-party authentication tool to complement our own framework. And it worked great with one browser, but not with another. When the tool was finally modified sufficiently to work with the second browser, it broke the first.

The origin of this situation was the fact that the integrator had some kind of sweetheart reseller deal with this other tool's provider. So the wrong tool was picked for the wrong reason. This caused all sorts of problems and finger-pointing until the integrator was finally pushed off the project. One of the first things we did when they left was replace that authentication tool. Nobody complained, because there had never been an internal champion for it.

Pick the right tools and the right people for the right reasons. Don't start compromising or you will lose your way. Sure, there are always political considerations, but if you don't focus on the ultimate goal, and remind your self of that goal regularly, you may end up off course.

Determining Specific Requirements

Zen and the Art of Motorcycle Maintenance is a tough read. When you get to the end, you'll say it was worth it, but getting there isn't easy. But I took away some great lessons on writing and creating from that book. One of the most useful of those lessons (and the timing was perfect as it was just as I was transitioning from typewriters to electronic keyboards) was *not* to try to make something perfect the first time through.

Years ago, there was a cereal commercial where a guy sits on the deck of his cabin preparing to write his Great Novel. So, he sticks the paper in his Olivetti and starts typing, Chapter One. But it doesn't work that way.

In *Zen*, the author teaches you to *make a mess and then clean it up*. Before you rearrange your hall closet, you pull literally everything out of it, figure out what's staying and what's going, and only then do you put things back. Otherwise, if you only take half of your stuff out, you shift things around and it's worse than before. You have to shake things up.

So when planning your IAM project, put everything on the table. You may not actually destroy everything and rebuild from scratch, but put it out there. Make an inventory of your wants and needs, the stuff that works and the stuff that doesn't, the stuff you've never had but really need, the stuff other people (like your auditors) keep telling you that you need, and the stuff you don't absolutely need but would like to have.

It's only a wish list, right? So make a mess, and throw the kitchen sink in. A lot of wishes will never make the final cut, but nothing is set in stone to begin with.

The Essentials of IAM

Let's make a mess by tossing around the IAM functions (and these are not necessarily IT functions) you need to consider:

- **Source of truth or authority on user identity and privileges** Where and how are you storing everybody and everything? Everybody knows they need a user store. But you also need to inventory your IT assets and their attributes. Will this data be kept in a single store, or distributed? If it's distributed, how will you aggregate it when you need to (and you *will* need to)?
- **Roles and their relationships to access** How do you want to define and use roles? Remember, you should end up with fewer roles than people.
- **Registration, by user, by manager, by import** How will new users be introduced to the system? For that matter, how will you get existing users into your new infrastructure?
- **Self-service** What functions will you allow users to do for themselves, and how will you expose those functions?
- **Requests and approvals** Who can make requests and how? Can users make request on their own behalf? Can managers request access for their employees? How do you want to notify managers that they need to take actions to enable users under their control?
- **Provisioning, re-provisioning, de-provisioning** What will your policies look like? Remember, you want to automate as much provisioning as possible. It's more sustainable, and reduces your manual processes. You need to get users on board; you need to manage change when they, their roles, their attributes, their groups, or their place in the organization is modified. You need to get them *out* as quickly as possible. And you need all of this audited, logged, and recorded. Who did what and when? Who requested it, who approved it, who actually executed it?
- **Integration between security apps and business apps** IAM enables users for the business now, and governs access to it later.
- **Authentication methods (what will make you feel secure?)** What credentials, PINs, tokens, security questions, biometrics, or charm-and-good-looks will make a user acceptable?
- **Authorization/access management** What sort of mechanism or policies do you want in place to match up assets and authenticated users? Where will the information that you need in order to determine if somebody is eligible to access a requested resource reside?
- **Coarse-grained vs. fine-grained enforcement** How deep in the weeds are you willing or required to go in enforcing access (“this group can access these pages” versus “individuals with this value in this attribute can view this account”)?
- **Compliance and how it's built into the system** How will your framework enforce regulations *before* bad things happen?
- **Reporting and audit support** How easy will it be to get comprehensible information out of the activity later? Do you merely need to report on what happened, or do you need to review types of activity, spot trends or reassess policies, risks, and privileges?

- **A good end-user experience** Happy users hassle you a lot less. They speak well of the organization. They back you up when you change policies. They don't e-mail your bosses demanding your head. They cut you slack during outages or upgrades.
- **Business partner requirements** Even if your company doesn't face certain regulatory requirements, your partners may, and therefore you can inherit those requirements. In addition, if you ultimately want your systems to interact with those of your partners, there's a whole other layer of integration, perhaps even authentication and authorization, that we'll talk about later.
- **Industry-specific needs** Does your particular market labor under specific provisions? There are casual compliance guidelines (meaning suggestions) and strict laws (meaning absolute requirements) that target particular industries. These include insurance companies, manufacturers, drug companies, healthcare providers, higher education, and so on. We will discuss these in more detail throughout various upcoming chapters.

Governance by Committee

This is also known as “Death by Committee,” because too many decision makers without common ground can bring a needed process to a halt. Everyone has an opinion, and you will hear each and every one of them. But while painful, and occasionally tedious, it is also necessary. You may absolutely know what you want and what the organization needs. You may be the smartest person in any room, the moral authority on the subject. Everybody else in the company might look to you and say, “We're all idiots, please lead us to the promised land.” And still you will form at least one committee to beat up any and all suggestions that arise, and to lend weight to the results.

None of us is as smart as all of us (although all of us are slower than any one of us). It takes a village. We're all in this together. Whatever clichés you want to employ, they all apply. You need multiple eyeballs on everything. It might all be the same actual group (although chances of that aren't great), but you will need to fulfill multiple functions, which likely means multiple groups of helpers.

Still, remember this important point: Every quest requires a leader. If you're that leader, then lead. Don't let the chair or emotional leader of any particular committee sidetrack you or the project. Be in charge. But at the same time, take advantage of both the wisdom that may come from the unwashed masses, and the cover provided by the forest of bodies.

It helps to delegate a little. Assign tasks. Give responsibility. Even if you don't mean it, sell it. People will perform better for you, and certainly behave more nicely, if they feel empowered.



NOTE

It's not just about software. As you're engaging the various groups, it's about their business processes. With regard to their place in the widget-making business, what goals do they need to achieve? What responsibilities do they have? What tasks must they complete? The software's just there to serve the goals. Think “process” at all times.

Committee rule often has bad connotations, as it conjures up images of some important matter getting bogged down in endless discussions and paralysis-by-analysis. But when you're talking about something this potentially large and *strategic*, you absolutely must sanity-check the big issues.

A Preposterously True Story

Here is how not to plan an IAM project. Just before the new millennium, a large manufacturer wanted to build, deploy, and secure an international supply chain portal. The customer's project manager chose his vendors for directory, database, routers, servers, portal, and access management (my company) based strictly on demonstrations. His design methodology consisted of gathering all the vendor reps in one room at the same time for half a day and having them cook him up an architecture.

Naturally, none of the vendors saw a problem with this approach, since they were all fairly guaranteed to get a purchase order. In this room full of happy people, I was the troublemaker, asking about a project plan, an integration map, a list of deliverables. The project manager replied that the salespeople in the room were going to provide him with that blueprint for success during this half-day meeting.

It gets worse. I made the mistake of asking what his timeline actually was. He said he wanted to go live in two months. "Let me get this straight," I said, "You're going to purchase, take delivery of, install, build out, and integrate all the software and hardware, populate the directory, install the database, set up the routers in front of the servers, build your portal, set up the auth schemes, and go live in 60 days?"

The rep from their integration partner spoke up at this point and insisted, "We know it's aggressive, but we think it's workable."

By now, I'm already unpopular with the other vendors, and since we're all going to get sued anyway if we go along with this insanity, I figured, what the heck. "If you had all this stuff already installed today," I explained, "you would spend the next two months just testing. Aggressive is not the word."

Apparently because we were the only ones speaking the truth, my partner and I became the trusted advisors, and in fact got the first actual purchase order. We went live against some existing applications, while some of the other vendors were dropped. When the original manager who brought us in was moved off the project (no surprise), I ended up with a very good relationship with his successor, who called me regularly for years even about unrelated technology. The integrator was also replaced.

This is an absurd (although true) example, of course. Nobody should commit budget and energy to a bag of air. You are responsible for your own destiny. Vendors want to sell software; consultants want to generate billable hours. That said, everybody should want you to be successful, because if you fail, you're a lousy reference.

But it's your project, so lead the way. You may engage third parties to help (and in fact you likely will, for some portions), but you're on the hook. You have to tell people what you want (as well as what you'd like to have, and remember that these aren't necessarily the same things), and in order to do this, you need a plan.

If *you* are the advisor, tell people the ugly truth. And if they're going off a cliff, don't go with them. When that project fails, your name is attached. Better to know the tough stuff up front and avoid going down a ridiculous path than take a chance that has, well, little chance.

Engage Stakeholders

Involve the actual line of business owners. If you're in charge of implementing security, chances are you're not a stakeholder, a line-of-business owner, someone with something to gain through an IAM framework, or someone with something to lose by sticking with business as usual. So you'll need to pull in those people to see where they may feel security is lacking. As you compile their wants, be sure to provide two important elements:

1. **Guidance.** Let the stakeholders understand that you're building a framework, not their own personal security system. They will be part of a larger picture. Tell them you will attempt, as time and logic and budget allow, to incorporate their needs into those of the entire organization. Without the guidance of this wish list, you will die from scope creep. Be prepared if, once you start building a list, interested parties try going over your head to force you to add their pet functions to the list. Part of this guidance ought to be anticipating what kinds of wishes you're going to get from the populace, so that you can have a notion of how you're going to accommodate them (if possible) in advance. Nothing sets people at ease, and keeps them off your back, better than proactively demonstrating that you're on the case and already aware of what they need.
2. **Proper expectations.** Tell them early and often that they won't get everything they want. You will need to prioritize the bigger picture, so tell them to prioritize their *smaller* picture.

Which Committees to Assemble

Bear in mind that a lot of these committee structures will come in useful again when it's time to actually deploy a solution. They will be helping to oversee, test, validate, and ultimately govern what comes out of a deployment.

So here are the different group initiatives or committees you may want to consider.

Executives and/or Stakeholders

Depending on how big your organization is, this might be the same group. These are the same folks who presumably signed off on the work, the costs, the budget, the requirements, the staffing, and so on. Make sure they stay on board with you, mentally, physically, and financially. If you're very, very lucky, you'll only have these kinds of committee meetings to update them on your wonderful progress. If there are any hiccups, these are your opportunities to remind the bosses that nothing is guaranteed in life and you're on top of it in any event.

Resource Governance

Who will represent the various IT assets that will be folded into the new IAM system? Somebody must own or speak for those assets. There will be integration requirements, perhaps some customization, and certainly the need to determine how those assets will communicate with the new IAM pieces. It's also a good time to ask what these resource owners will require later in terms of access review (for example, attestation).

Role Governance and Segregation of Duties (SoD)

I lump these together because, while duties are often segregated at a very granular level within individual applications, the goal of SoD is to not concentrate incompatible privileges in the hands of one individual.

Role definitions play a huge part in provisioning. They serve as the bridge between the business and the technical aspects of user enablement. The business says, “You’re an accounts receivables clerk,” and that translates into “You get a data entry slot in Peachtree, reporting rights in Oracle Financials, this set of menus in Siebel, and this set of reports in the CRM package.”

What goes into a role from the top to the bottom has to be agreed upon, in a management sense, from the top to the bottom. HR and/or upper management decides what jobs need to exist just to run the place, middle management decides what actual job functions go into each position, and IT helps map the IT assets to those job functions.

If and when you perform role discovery on existing user-privilege data (something we’ll discuss in functional detail in Chapter 9), what you end up with are resource-level roles, or application/resource level roles. These are the contents of a business or enterprise role. To reiterate, a job title or position is the equivalent of a business role. It gives you access to all the resource-level roles you need in order to do your job. Each resource-level role contains all the little privileges needed to execute that role.

If you’re enforcing privilege-level SoD, there’s a chance that nobody on your committee will have the information they need to create those policies. If you produce or purchase a library of these, and they exist for a number of off-the-shelf packages, then you’re pretty well set. Recreating those kinds of libraries is not a fun business. Sometimes they come as part of an enforcement package, but otherwise you can typically provide them as source data for your provisioning or access control software.

If you’re going to enforce SoD at a higher level, all the better for you. It’s far easier. But be aware, as much as this idea is beloved, and is even supported by the National Institute of Standards and Technology (NIST) standard for RBAC, it won’t necessarily get you through an audit. Find out what your auditors will insist on.

If your IAM software supports it, it’s a great thing to associate roles with branches or individual nodes of the organization. Not only does this help with automatic role grants and approvals, but it also lets you take two possible approaches on higher-level SoD. If you have a position down one branch, it prevents you from having one in another branch. And whatever software you run, it had better support role grant policies. Within these policies, you should be able to specify inclusion and exclusion rules. To have a role, you should be in a particular group or branch or line of business. Or perhaps to have a role you must specifically *not* be in a particular group or branch or line of business. So these policies that I keep harping about should help you create prerequisites and also walls of functional segregation.

Audit/Compliance Requirements

Your financial department is probably a good place to start when recruiting for this group. Compliance really starts with the money. But it certainly doesn’t end there. You may have industry-specific regulations that must be addressed via IT security. Insurance, healthcare, higher education, and a host of other arenas of special interest have their own requirements.

Audit support too often gets put off until you’re scrambling during that first post-IAM audit. You can’t get the reports out of it that you need. Or those reports are all custom. Don’t let the vendor or partner slide by with a wink and “Oh yeah, we can squirt that out later.” Ask to see examples. One of the first? Attestation. But remember, with a proper provisioning tool, attestation isn’t just a report; it’s a process with a report that comes out at the end. It shouldn’t *be* just a report.

Of course, your outside auditors, just to show they’re doing their job, *will* ask for custom reports, so make sure your report writer is up to the task. There’s nothing sneaky or illegal about

going to your auditors, asking what they'll want to see, and just plain telling them that you're putting a new IAM framework in place.

Roll-Out Plan

This committee will actually have two functions. First, planning a phased rollout and sanity-checking it with the rest of the crew. They need to devise a reasonable plan. The other function is devising a communication plan that will keep the organization in the loop as the rollout is taking place.

Let's start with communications. There will be hiccups and surprises. There will also be a lot of good stuff. By keeping the corporate populace informed, you will insulate yourself to some degree when the inevitable mistakes do occur. You're also providing named personnel who can field questions. Set up an internal blog, an e-mail list, a landing page, or all of the above so that everybody knows what's going on, what will happen and when. Let them know the proper channels to use when they have questions: call this number, click this link, e-mail this contact.

Okay, so there's the communication of what you're doing. Now, you must do it. While it might be fairly simple to place the entire organization under a basic authentication bubble, it might take a bit more to get everybody integrated with the single sign-on umbrella. You have to teach SSO how to provide the necessary attributes (quite often header variables) to each application following the initial authentication. But even then, SSO is one of those low-hanging fruits.

A *huge* part of rollout planning is deciding which resources get brought on board and in what order. You may decide on some combination of resources that

- Are most critical to the business
- Are SOX-based
- Have already been breached
- Are owned by key stakeholders or budget masters
- Have the most users
- Process the most daily transactions
- Are customer-facing
- Are managed by staff who have the cycles to help

Cut-Over

Yes, yes, you're rolling out in phases. But for each individual component, there's a single nanosecond in history when you move from no-framework/old-framework to new-framework. You throw the switch and *bang*, you're now authenticating against the new web server plug-ins and new directory and so on. For new applications being launched, you have the opportunity to beat them up in a test environment. Yes, you can do this with the existing resources as well, but now you're talking about people in production suddenly being thrust into a new scheme.

For the last large cut-over I worked on, we kept a couple of dozen willing volunteers behind on a Friday evening, performed the necessary actions to move from one platform to the other, then had people sit down, log in, and go through several use cases using recoverable production scenarios. Recoverable, in case something completely unexpected occurred. This was not the same as unit testing or load testing. These were real people. This was more nerve-wracking than the load testing with a hundred times that number of virtual bodies, because these volunteers

could point out our mistakes, real-time, and in what had now become the live, production system. There was one small glitch, quickly remedied, but all the hundreds of hours of prep paid off.

In a global operation, it's obviously a different story. But if you have segregated environments, or a scheduled blackout, it's possible to do this. There is *never* a good time for a shutdown, even a well-publicized, scheduled shutdown. Someone will always complain. No way to avoid it. With luck and preparation, it should be very brief.

Here's why a committee is a good thing here: planning the actual cut-over day. Consider this. There's Day One, and there's everything after. This is the Day One Committee. The really ugly stuff, as in loss of service, broken links, people who suddenly can't authenticate, people who panic because they haven't listened to all the information you've been broadcasting all along, all that stuff happens on that first day. Once you get past that day, it gets better. Even when things go smoothly, and with enough planning they will, things will also be *different*, and all the individual users who didn't take advantage of your rollout blog or emails will be in the dark. Nobody listens to the flight attendant when she's explaining how to escape in the event of a water landing, right? Nobody pays attention to the first five e-mails warning you to change your password before it expires, right? Same deal here. The Cut-Over Crew will remind people as the fateful day approaches that it is indeed coming, and they will be there to hold the hands of all the idiots who ignored them.

An Iterative Process

This approach requires multiple passes. As you have your meetings and get your committee feedback, you will revise your plan and your ideas. You will reprioritize multiple times. You won't get this right the first time through.

No matter how much you plan, something will go wrong. It's inescapable. But the more you plan, the less will go wrong. And the more you communicate, the less people have a right to complain. Communication is also a two-way street. You put out information, and you get opinions in return, and some of them may be quite valuable. You don't know what you don't know.

If basic name and password are not sufficient for all lines of business, then you will not get instant agreement on a common authentication scheme. And IT professionals who are used to doing everything based on LDAP often feel that group memberships are sufficient arbiters of authorization; others depend on attributes, organizational placement, and relationships, while more sophisticated approaches employ roles. The point is that consensus is an elusive goal, requiring the revisiting and negotiating of the ultimate solutions.

In Chapter 5, we'll discuss in greater detail how to make the ultimate business case to management, that is, explaining the value of IAM to the enterprise. Once again, having at least the semblance of a plan will provide the appearance, and the structure, of organized thought. If you get the go-ahead, then this planning, your committees, your lists of must-haves and wants, will form the basis for your actual deployment plan. It would be nice if you could just get to where you're going. But you need a map, and gas money.

