# Big Ten Academic Alliance Provisioning Cookbook

**Document Title:** Big Ten Academic Alliance Provisioning Cookbook

**Document Repository ID:** TI.166.1
**DOI:** 10.26869/TI.166.1
**Persistent URL:** http://doi.org/10.26869/TI.166.1
**Authors:**

- Omer Almatary, Rutgers University, https://orcid.org/0000-0002-6704-7052
- Erik Coleman, University of Illinois Urbana-Champaign, https://orcid.org/0000-0002-5952-0946
- Beth Halsema, Purdue University, https://orcid.org/0000-0003-3061-0961
- David Halsema, Purdue University
- Keith Hazelton, Internet2, https://orcid.org/0000-0002-3174-899X
- Karen Herrington, Virginia Tech, https://orcid.org/0000-0001-8691-2358
- Ethan Kromhout, UNC Chapel Hill, https://orcid.org/0009-0006-5306-742X
- Gail Lift, University of Michigan, https://orcid.org/0000-0001-8282-4019
- David Walker, Consultant, https://orcid.org/0000-0003-2540-0644
- Keith Wessel, University of Illinois Urbana-Champaign, https://orcid.org/0000-0002-8047-3187
- Richard White, Purdue University

**Publication Date:** May 2023
**Sponsor:** Big Ten Academic Alliance

# Big Ten Academic Alliance Provisioning Cookbook



## 1. Introduction

Provisioning is a core function of any identity and access management system. Whether you're provisioning identities for new students, MFA devices for staff members, e-mail boxes for faculty, or secure cloud storage for a research group, there's lots to think about. Each kind of item that you provision has its own set of considerations and pitfalls. It's important to have standard practices and policies around your provisioning logic so that things flow smoothly and everyone knows what to expect.

Some provisioning practices, like deciding at what point your users have their e-mail accounts created, is easy to adjust as you go along. Other things like defining a useful and extensible set of roles to determine what users can access is much harder to adjust once in place. Regardless of what part of your identity infrastructure you're laying out, careful thought and planning up front can save a lot of headaches later.

Another aspect that goes hand-in-hand with provisioning is auditing. Now more than any time in the past, auditing of systems for compliance with legal regulations is a huge priority. Sometimes, however, it’s just as important to audit systems to make sure that local institution policies are being properly enforced. Doing this manually, though, is next to impossible. Things work best when your auditing and reporting mechanisms are built into and work alongside your provisioning systems.

This cookbook seeks to jumpstart your planning with easy-to-follow advice and trusted practices that others have found work well. It's written from the perspective of provisioning in a higher-ed environment, but the advice applies in other settings, too.

We've created this cookbook with a simple, straightforward format. The first section defines concepts and terminology that's helpful to understand before diving into the specific recipes. The sections that follow each cover a specific area of provisioning and deprovisioning. Each section starts out with an overview of that concept, followed by the specific guidance.

We offer our guidance in “DO”/”DON'T” form: each item starts with “DO” or “DON'T”. Items that aren't necessarily a definite DO or DON'T but that are worth keeping in mind start with CONSIDER.

There's no such thing as one size fits all in provisioning. Much of the architecture depends on your organization's needs, culture, and size. Institutional policy will be required (and, perhaps, evolved) to shape IAM business processes and system architecture. While this cookbook can't tell you exactly how to architect everything, it explains best practices and the concepts needed to get the job done. Our goal is to help you understand what you need to know so that you can make your own informed decisions appropriate for your organization.

## 2. Problem Statement

When it comes to identity management, higher-ed institutions are complicated places. A single person can have multiple roles all at once: student, alumnus, staff member, and parent of a student, for example. That same person can disappear from any active affiliations with the institution, only to reappear several years later with a whole new set of roles. Managing the digital identity, credentials, services, and roles of that individual is no small task.

Consider further that higher-ed institutions on-board hundreds or thousands of new users each year as new students enroll. This means that, not only do the policies and practices that a school has in place need to be well-defined, but the systems that enforce them need to be scalable. Institutional roles need to be able to express affiliations concisely and accurately so that systems can easily perform the work to create all of the resources that those students will need to learn and participate.

Then, every spring, hundreds or thousands of students will graduate. Those same systems will need to deprovision resources. Sometimes, some of the resources will be maintained if the school offers continued services to alumni. Other pieces of information will be maintained to create an auditing trail or to allow easier on-boarding if one of those students comes back later for another degree or a job.

In addition to the usual identity lifecycle of students, faculty, and staff, most institutions see a large number of guest or sponsored accounts. Visiting scholars, contractors, K-12 students attending summer camps – they all need a digital identity while they’re interacting with the institution. Unlike longer-term engagements like that of a faculty member or student, though, knowing when to provision and deprovision these users can be very challenging. Regardless of the user population, how do you know when specific users need to have access and what services they need to access?

With so much data flowing, being able to audit the full system and make sure that nobody is being incorrectly given access is crucial. A good provisioning system also has good auditing capabilities and good reporting functionality.

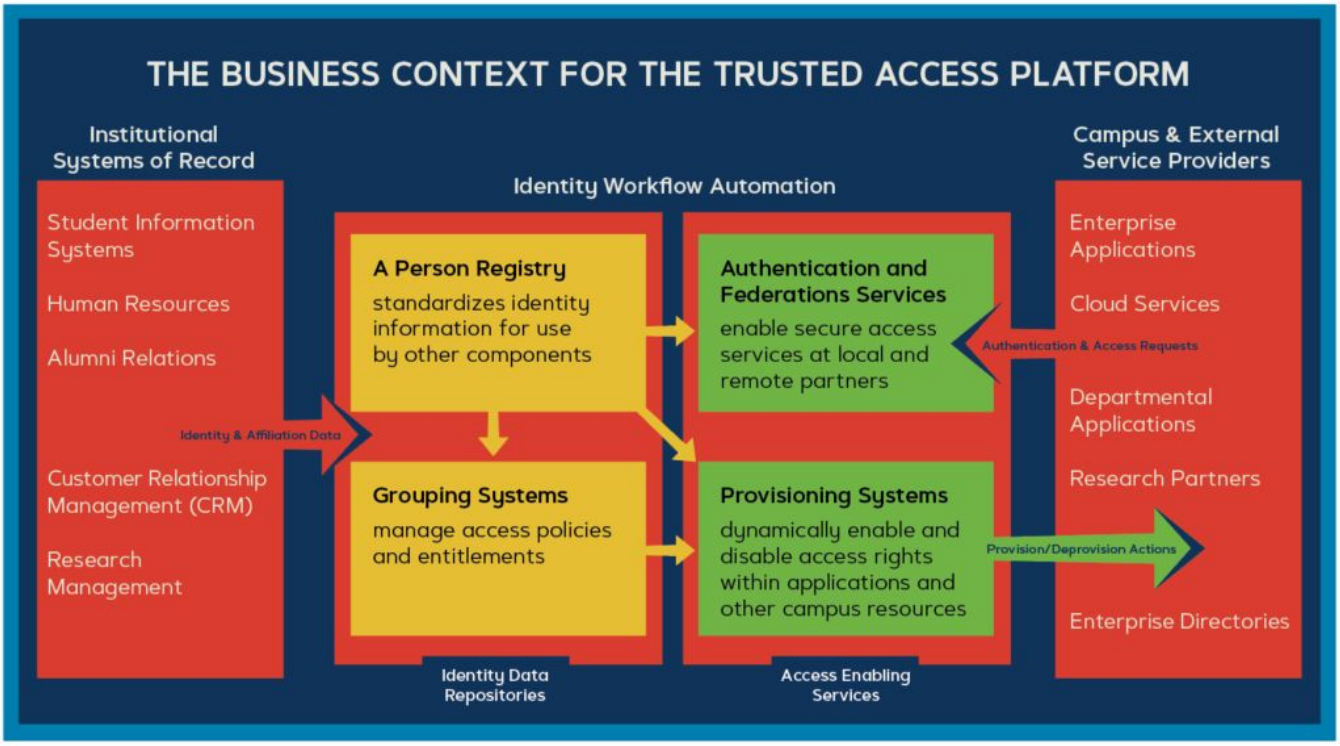
Once all that's done, you shouldn't overlook deprovisioning. Especially in these days of cloud-based services and per-user licensing, failure to deprovision a set of users who no longer need service can become costly over time. When you deprovision and what you do with the data in identities and services before they're deprovisioned depends on your exact situation. For some cloud-based services, you might even choose to let your users convert their accounts to personal accounts that they can continue to own but that are no longer affiliated with your institution. The important thing is that these decisions are made up front.

Obviously, the big picture can quickly become overwhelming. There are lots of things to consider and lots of decisions to make, each one that will significantly impact your users directly. We'll tackle this challenge one piece at a time.

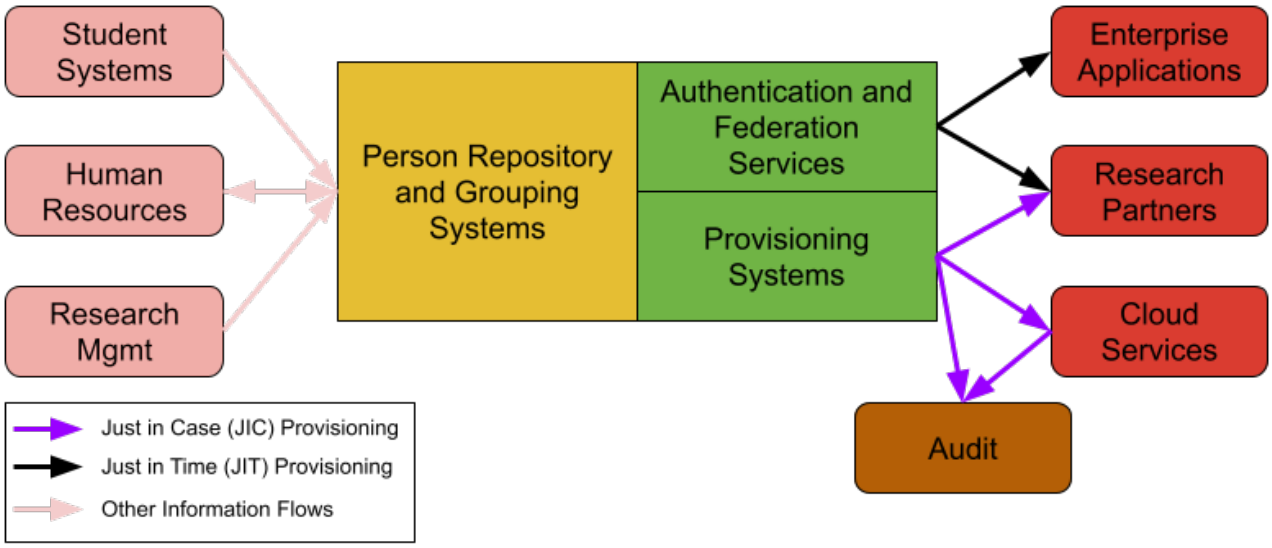
## 2.1. The IAM Business Function and the IAM System

Addressing these issues is generally addressed by an organizational unit within the institution, usually central IT. In partnership with other units, such as Payroll, Registrar, Library, research projects, student organizations, *etc*., the IAM unit is responsible for the business functions that maintain the institution's unified repository of its community: students, staff, faculty, library patrons, researchers, members of the chess club, *etc*. The IAM unit is also responsible for the controlled dissemination of selected information from that repository to online resources to facilitate access control and personalization.

In order to accomplish this for the constantly changing hundreds of thousand members of the community, IAM Systems are used to automate the lion's share of these processes. The following diagram shows the interrelationships among affected organizational units and components within the IAM:




This document concerns itself primarily with the provisioning of data out of the IAM system, but also with flows from Institutional Systems of Record, as highlighted in the following diagram:




Provisioning flows out of the IAM system to service providers facilitate access control decisions and personalization by online resources, providing those resources have the information they require to make those decisions.

The provisioning flows out of the IAM System can be classified as:

- "Just in Case" (JIC), flows that occur in case one of the recipient services needs the information at some time in the future, or
- "Just in Time" (JIT), flows that occur at the moment the information is needed by the recipient service, generally as part of an authentication event.

For example, when identity information is transferred at the start of a session with a Service Provider, it is JIT provisioning, generally accomplished by the authentication system *(e.g.*, as a SAML assertion or a collection of OIDC claims). When identity information is transferred as the result of some other event (*e.g.*, a change generated by a System of Record), it is JIC provisioning.

Audit systems also receive provisioned information from the IAM System to assure that access control policies are consistent with reality, Note that audit systems may also retrieve information from the service providers to enable comparisons between the service provider and the IAM System to find discrepancies.

While only secondarily within the scope of this document, information flows between Systems of Record and the IAM System may be JIT or JIC flows, driven by changes in state within the systems of record, to keep the IAM System's Person Repository and Grouping Systems current with data about community members from the Systems of Record operated by partner organizational units, such as Registrar and Human Resources. Note that these flows may be bi-directional, when the IAM System holds information of use to a System of Record. In this case, such flows are driven by changes in state within the IAM System. Examples of such information include user names and directory information.

# 3. Basic Concepts

## 3.1. Identity and Subject

In the context of the field of Identity and Access Management (IAM), *identity* refers to the set of information that pertains to a *subject*. A subject is typically a person but may be an institution, a well-known service, *etc.*; it is the thing with which the identity is associated. The information that comprises an identity may include identifiers, group memberships, entitlements, roles, names, and other characteristics.

## 3.2. Identifiers

*Identifiers* uniquely distinguish an identity within a given *domain*. Typical domains might be a university or other organization, or a federation of universities and other organizations. Examples of university identifiers might be called netID, login name, principal, or username and have values like s `usan.smith`, `jrrtolkien`, or `a983k50299`. Electronic mail addresses may also be used as identifiers, although the practice is discouraged, as electronic mail addresses may change, may be reassigned to other people, may be used by more than one person, *etc*. In summary, it is usually undesirable to manage electronic mail addresses in a manner commensurate with their use as user identifiers.

*Federated identifiers*, such as the SAML General Purpose Subject Identifier (subject-id) or the eduPerson Principal Name (ePPN), are unique within their federation and are typically structured by appending a unique institution identifier to a person identifier that is unique within the institution. For example, if John Doe's identifier within the University of Illinois is `jdoe`, then his federated identifier might be `jdoe@illinois.edu`.

(See Understanding Federated User Identifiers for more information.)

### 3.2.1. Uses for Identifiers

Identifiers are used in multiple ways:

- Internally to link identities among multiple services and directories
- Externally with federated vendors
- By the user as part of an authentication event
- To retrieve attributes or other information associated with an identity

### 3.2.2. Types of Identifiers

Identifiers have multiple characteristics.

- They may be usable by any service provider, or they may be *pair-wise*, generated uniquely by the identity provider to be usable by only a single service provider.
- They may be *human-readable* to be easily recognized when read, or they may be *opaque*.
- They may be *short-term*, for example for the lifetime of a session, or they may remain valid for longer than a single session, always unique to the same identity. How long an identifier remains valid is also an important characteristic. Identifiers are considered to be *stable* or *immutable* if they remain valid over a very long period of time (*e.g.*, longer than the existence of the identity within its domain).
- An identifier may be re-assignable to a different identity, or it may not.

It should be noted that identifiers that are used to control secure access to resources or services should be stable and not re-assignable to mitigate unintended loss of access by an otherwise authorized subject, as well as unauthorized access by a subject who receives a re-assigned identifier. Also, unless there is a need for multiple services to coordinate (*i.e.*, track) their offerings for the same subject, pair-wise, opaque identifiers are generally preferred to enhance privacy.

More information regarding *federated* identifiers can be found in Understanding Federated User Identifiers and Choosing the Right Federated User Identifier.

## 3.3. Affiliations, Roles, Groups, and Other Attributes

There is a rich set of information that can be used to determine a person's permissions to use services and resources.

- *Affiliation* identifies how or why a person is a member of the institution's community. Examples might be student, staff, or faculty. A person can have multiple affiliations simultaneously.
- *Role* identifies what a person does for or with the institution. Examples might be instructor, principle investigator, or IAM administrator.
- *Group* is a collection of people that share a common characteristic. Examples might be people with the instructor role, members of the chess club, or Physics 101 students.
- *Attribute* is a broad term for just about any information associated with a person. Examples are telephone number, electronic mail address, or job title. Not all attributes are useful for assignment of permissions, but some are.

## 3.4. Provisioning Models

There are two types of provisioning, *Just in Time* and *Just in Case*.

- *Just in Time* (JIT) provisioning occurs when identity information is provisioned at the time it is required. Just in Time provisioning is usually implemented as part of (or immediately after) authentication for a session.
- *Just in Case* (JIC) must occur before the identity information is required. Just in Case provisioning occurs as the result of some external event, such as enrollment of a student or assignment of a new role to an employee, or as regularly scheduled processing, such as overnight reconciliation between the IAM and a service.

It should be noted that there are gray areas here. For example, if a student must visit the help desk to gain access to a service, in order for their account on the service to be created, it may look like JIT to the student but JIC to the technology.

# 4. Identity Provisioning

At universities, it is typical for Admissions and Registrar to determine whether an individual's affiliation with the institution is as a student, Human Resources determines whether that individual's affiliation is an employee, and other organizational units to determine other affiliations. It is also possible for an individual to have multiple affiliations. For this reason the IAM function needs to form partnerships with those other organizational units to obtain the data of who, currently, are students, employees, *etc*. Operationally, this is implemented by identity provisioning interfaces between each of your partners' Systems of Record and the identity registry. An identity matching algorithm is then used to determine when an individual having multiple affiliations receives records from multiple sources.

## 4.1. Identity Matching

Many organizations, particularly universities, have multiple source systems representing multiple segments of the population. These include staff, faculty, students, parents, physicians, friends of the library, *etc*., to name but a few. These segments, however, are not mutually exclusive. A student can be a staff member, for example. For this reason, records retrieved from identity source systems must pass through an Identity Matching algorithm to ensure that a) one person does not appear to be two people in your IAM system, and b) two people do not appear to be one person.

### 4.1.1. Do: Use a scoring system that separates new identities into positive matches, unmatched, and potential matches.

When creating a new electronic identity, it's important to see if the individual for whom you're creating already has an electronic identity at your organization. Not checking carefully can lead to someone getting two identities or, worse yet, incorrectly assuming that two people are the same person. A scoring system works well to solve this challenge. Compare common elements of the new identity with those already in the system such as date of birth, gender, name components, and whatever else you have available. An exact match might get a high score unless it's a common name: there could be two individuals named Elizabeth Jones both born on March 1, 1980. So, common names might score lower. Use all of the information you have, score each comparison intelligently, and decide how high of a score counts as a match. Where practical, engage directly with the person being processed through self-service processes to verify knowledge of data you already have, such as student ID, course information and grades, employee ID, *etc*.

You may want to consider categorizing certain data elements used in the matching as High Assurance, Medium Assurance and Low Assurance. For example:

- High Assurance: IDs from systems of record, the registry's unique ID, SSN, driver's license, state ID, passport, NetID
- Medium Assurance: name, date of birth
- Low Assurance: email, address, phone

These are, of course, only examples; you will need to determine each of your data elements' contribution to the matching algorithm, based on the characteristics of your population and the quality of the data you receive from the various sources.

Note that, once a positive match has been determined by the scoring system, you should collect the System of Record's identifier, if possible, so that the scoring process does not have to be applied a second time.

### 4.1.2. Do: Establish a process for putting potential matches in a suspense group for manual review and reconciliation.

It can be easy, using the above system, to know what to do if you definitely have a match or if you definitely don't have a match: high and low scores can be handled automatically. But what about scores in the middle? This is where a good workflow process comes into play. Create a queue for the "maybe" cases, and assign staff to review it manually to research if a new identity is the same as an existing one or really a different individual.

### 4.1.3. Do: Establish a process to correct mistakes made either in the automated or manual identity matching processes.

Mistakes are unavoidable. You may not detect them until long after identity assignments have been made.

### 4.1.4. Consider: Products and processes that facilitate matching such as phonic and fuzzy name matching, and address standardization.

Non-exact names might still be a match: a new identity for Beth Jones might be for the same individual as the existing identity belonging to Elizabeth Jones. Also consider that data entry can contain typos and a name could have an error in it because someone entered it wrong. Using software that can do non-exact comparisons of names or look at phonic comparisons will help immensely in identity comparisons. Similarly, changing addresses to a standardized format where geographically available can give another clue if someone is a match.

If you can, engage the actual person whose new data is being processed to participate in a self-service to match with the existing data you already have on the existing record such as Student Id, Course Information and grade , Employee Id. Something similar to credit card report validation.

## 4.2. Institutional Username Assignment

Your users' institutional usernames (also known as netid, login id, or userid) are the unique identifiers they use for your single sign-on system, facilitating access for many resources. It may also be used for institutional electronic mail addresses, although many institutions do not do this, as it reveals part of what is needed to authenticate. It is also the case that service providers may or may not use the institutional username 1) because the institutional username is not in a format that the service provider can use, or 2) as with electronic mail addresses, it may reveal information that is better kept secret.

### 4.2.1. Consider: Self selection *vs.* assigned usernames.

There are good reasons to automatically generate usernames for your users. There are just as many good reasons for letting them choose their own usernames. It's up to each institution to do what's best for them, but consider the pros and cons carefully. Letting users pick their own username generally leads to nicer looking and easier to remember usernames than an algorithm can select, but there's always the risk that a freshman might pick a questionable username. When that freshman later discovers that "it seemed like a good idea at the time", you're opening yourself up to more username changes later. Some institutions have had no problems with students choosing their own usernames. Others choose to generate usernames for students while letting faculty and staff choose their usernames. Regardless, if you're going to assign usernames automatically, think carefully about your algorithm for username generation to try and maximize the use of your username namespace with reasonable looking usernames. Nobody wants a username that's an unpronounceable string of consonants followed by a

random-looking string of numbers, especially if it's also part of their institutional email address. It's also possible that a generated username could have cultural sensitivities or be otherwise offensive to an individual. So, if you do choose to use assigned usernames, consider having a documented exception request process and using a vanity/alias for the email address.

### 4.2.2. Do: Allow changes for good reason.

Though generally painful finding all of the provisioning and authorization implications in downstream systems, username changes are inevitable. Marriages, adoptions, and other life events that result in name changes are often good reasons to change a person's username. E nsure you have a process that preserves the previous username so it cannot be reused by another user, and keep as much information as is practical on downstream systems that may have a copy of that username provisioned to them.

### 4.2.3. Don't: Reassign a username to a different person.

> While it may seem tempting, especially in a space where all the good usernames are already taken, to give the username of a long-graduated or long-retired user to a new student, there are very good reasons not to do that. There's the obvious possibility that the original owner of that username might return. There are also resources and permissions that might still belong to that username which you wouldn't want to transfer inadvertently to a new person. Non-reassigned usernames are of particular importance when dealing with cloud services that might still have data belonging to the existing username. In general, assign a new username instead of reusing a dormant one.

### 4.2.4. Do: Make sure the namespace is large enough to not run out for many years.

This is common sense, especially given the practice of not reassigning usernames. The question is how large is large enough? That depends on the size of your institution and the number of users you're on-boarding. There's no magic formula for how many characters long you should allow for the shortest and longest usernames and what characters (letters, numbers, certain symbols) you should include. You just need to find the right balance for your population. Naturally, usernames that are too long become unwieldy. Usernames that are too short lead to indecipherable strings. The right length and, if appropriate, the right username generation algorithm can go a long way, and your future self will thank you. Potential strategies that can help include:

- including middle initials
- appending sequence numbers when there are conflicts (Beware, though, that some numbers have potentially negative cultural significance.)
- enabling self selection of usernames

### 4.2.5. Do: Check for a user's existing identity at the institution before assigning a new username.

There's no better way to burn through your username namespace too fast than to use it up assigning multiple usernames to the same individual. With institutions assigning usernames for guests, summer program participants, faculty, staff, students and many other populations, it's extremely common for a given user to pass through multiple affiliations, perhaps leaving and returning between each one. It's not only kinder to your username namespace to check for existing usernames before assigning a new one, but it's also kind to the user to give them back the username that they had last time they were part of the institution. See the information on identity matching for advice on how to best detect when a new user is the same person as an already existing identity.

### 4.2.6. Do: Have a list of usernames that should not be used.

Reasons for inclusion on the list include being misleading (*e.g.*, a username of "President"), being culturally inappropriate, or violating a trademark. The list would apply to both usernames that are assigned or selected by users.

## 4.3. Identifiers for Services and Target Directories

Some of your service providers will require you to provision a username to meet their specific requirements. Here is some advice.

### 4.3.1. Do: Maintain an opaque identifier that won't change over an entity's life cycle.

Users will want or need to change their usernames, and for many good, and some not-so-good, reasons. It doesn't take long to discover that a username as a stable identifier is anything but stable. If a user's identity is going to stay tied together across multiple directories and services, you'll need something better than a username to associate between those directories and services. The days of social security numbers for this purpose are long gone. Some institutions are comfortable using a university ID number for this purpose. Others choose to generate a random, reasonably long, unique string for this purpose when the user's electronic identity is created. Such an identifier doesn't need to be human readable or ever displayed to the user. The user doesn't even need to know that the identifier exists. It's purely for linking between systems.

### 4.3.2. Do: Consider an external identifier different than the internal identifier used by in-house applications.

There are good reasons to have multiple user identifiers for a given user, each one with its own purpose. While it's important to not create too many as that becomes challenging to manage, there are good reasons to create new ones. One such case is for external services and systems. Especially since these types of systems are more out of the institution's control, there's always the possibility that a bad actor could get your users' data. With the same stable identifier used everywhere, activities and data could easily be correlated between multiple systems. The best way to avoid this is to have a different identifier for a user for each service that they access. Rather than creating a new identifier for each service when the user is provisioned into it, consider an algorithm that could encrypt a string containing the user's internal

stable identifier and the name of the service they're accessing. The external identifier could then be built on-the-fly each time it's needed. This type of identifier is called a pair-wise identifier.

## 4.4. Social IDs

Most people already have IDs from one or more social media platforms. These can often be linked with institutional usernames in your IAM System to provide an alternative login method.

### 4.4.1. Do: Consider where account linking of social IDs to institutional accounts might be appropriate and where it might not.

Social credentials such as those issued by Google or Facebook can be a great choice for users who interact with non-regulated data. For example, alumni returning after a long period to request a transcript are much more likely to remember their Facebook login that they linked to their institutional identity, than to remember a username and password issued by your institution. Users are also quite likely to notice if their Facebook or Google credential has been compromised, which they won't for an infrequently used institutional credential. Common social identity vendors such as Google are very good at notifying users about changes to their accounts and credentials. Be careful though when using social credentials for regulated data as you lose control of credential quality, and social providers generally won't tell you if multi-factor authentication was performed, so you may not be able to guarantee some of the controls that are required by regulations.

### 4.4.2. Do: Consider whether social ID can be a step in onboarding/offboarding.

Social identities can be a great way to begin interacting in an authenticated session with users while you build up enough information to do identity proofing and other identity onboarding steps. Consider, for example, an applicant for a position in your HR system, it could make sense to use a social identity to allow the applicant to upload information during the application process, and even during background checks and other steps required before they are considered an employee in your HR system.

### 4.4.3. Do: Consider Level of Assurance LOA when using social IDs.

Understand the identification, registration, authentication, and related processes employed by the social providers whose IDs you use.

### 4.4.4. Don't: Assume all services do proper authorization and make them aware of the LOA concept.

Work with your service managers to assure they understand the risks and benefits of using social IDs for their service.

### 4.4.5. Do: Identity matching, even with social IDs.

It is fine for a user to have multiple social ID's, such as one with Facebook and one with Google. When registering those social ID's for use as a credential in your systems, keep track of the tie to a single institutional identifier. You don't want users trying to remember which social ID they used for which of your services.

# 5. Identity Lifecycle

Lifecycles of users in higher education environments are complicated. A given individual can appear as a high school summer camp attendee, an applicant, a student, an employee, a retiree, or any of a number of other roles. Further, that individual can have more than one affiliation at the same time. Keeping track of those affiliations and when they start and end is essential for providing and revoking access to services. The recipes in this section will help to define the parameters and processes needed to manage identity lifecycle.

## 5.1. State and Affiliation Changes

Many events, both large and small, affect identities and the access permissions associated with them. For example, a graduating student who is hired as an employee will gain permissions as appropriate to their new position and job responsibilities. That person will also lose permissions granted only to students. All of this is driven by information obtained from the source systems.

### 5.1.1. Do: Capture changes in affiliations/roles that matter for service entitlements.

The most important thing to do with user affiliations is to track when they change and to communicate those changes to downstream services in a timely fashion. Whether a user is being on-boarded, off-boarded, or just gaining or losing one of multiple affiliations, track this information.

### 5.1.2. Do: Work with service providers to ensure service entitlements are being handled correctly.

Once you're tracking affiliations, you need to help service owners to map those affiliations to specific service access. A service owner, for instance, might say that they want their service to be available to all students. But what does student mean? Full-time as well as part-time? On-campus as well as remote? Should they gain access to the service at the start of their first semester, or should it happen when they've registered for classes? Help service owners to understand all of the different transitions and options for understanding a given affiliation so that they aren't giving too much or too little access to their service.

### 5.1.3. Don’t: Overdo state changes.

If a state change doesn’t change an entitlement, it may not be necessary to distinguish it. A student, for example, who has registered but not yet started classes might need to be distinguished as some services might want to grant access before classes start. There are other cases, though, where state changes might not change any access such as a staff member changing positions within a given team. As mentioned above, carefully consider what you consider an affiliation so that you don't end up with thousands. There's a balance between future proofing your affiliations and information overload.

### 5.1.4. Do: Account for users with multiple overlapping affiliations.

This scenario is very common in higher education, but much less common in other organizations. Someone can be a staff member taking classes or a student with part-time employment. A retiree can come back as a student. There are lots of possibilities. Ensure that your system can assign multiple affiliations to an individual that can be separately assigned or removed.

### 5.1.5. Do: Designate a "primary" affiliation for each user.

Most services will care only if a person does or does not have a specific affiliation. Some services, however, need to know a person's primary affiliation with the institution. For example, a staff member taking classes might have a primary affiliation of "staff," but a student employee's primary affiliation might be "student." Also, eduPersonPrimaryAffiliation is useful in some federation scenarios. Institutional policy will be required to determine which of a person's affiliations is primary.

## 5.2. Grace Periods

Revoking access permissions may adversely affect your users, even when it is warranted. Grace periods can give people time to prepare.

### 5.2.1. Do: Make grace periods, subject to institutional policy for affiliations, roles, services, *etc.*

In most cases, though not all, you won't want an affiliation change to remove a user's access immediately so that access isn’t prematurely terminated. For instance, staff might need access to some services for a couple of weeks after leaving the university, and students might need to retain access for the summer after graduation. Building a grace period into these transitions can make for a better experience.

### 5.2.2. Do: Work with stakeholders to determine how long a grace period should last.

How long a grace period for a certain state transition should be varies depending on your organization, the state change, and the services that it impacts. In some cases, immediate loss of access won't be a problem (or may be required even if it is a problem). In others, it could lead to lots of confusion, extra work for service owners, and help desk calls. Consider each state transition and work with service owners to choose timing appropriately.

### 5.2.3. Don’t: Overextend a grace period if it compromises security.

It's easy to go too far on how long you leave things active. The longer an account stays active, especially if it's unused, the more likely that a compromise to that account will go largely undetected. Weigh this against user convenience when selecting how long access should be retained.

### 5.2.4. DO: Allow for immediate deactivation when necessary.

There will, of course, be cases where a grace period is not only unnecessary, but it's dangerous. The immediate termination of an employee or the departure of a high profile individual from the organization are often reasons to immediately remove access. Include a mechanism for bypassing the grace period when needed to immediately remove access.

## 5.3. Deactivation

Here is some guidance for when the grace period is past, and it's time to deactivate.

### 5.3.1. Do: Retain minimal data when deactivating an identity.

Just because someone's leaving your organization doesn't mean they're gone for good. Consider a situation like a student graduating, going elsewhere for grad school, then returning to take a job. For the purposes of identity matching to assign the same username or university ID number if available, or at least for building a complete history of a given user, it can be helpful to at least keep a stub entry with information about the previously active identity. What data is contained in that stub entry depends on your organization's environment and what you might want to track or reinstate.

### 5.3.2. Do: Establish policies and processes to reinstate disabled identities.

Just because you keep a stub entry for a disabled or departed user doesn't mean that re-activating the entry upon the user's return will be straightforward. Carefully consider how to most easily and efficiently turn the stub entry back into an active entry, confirm the user's identity, and possibly make sure that there are no lingering permissions granting them access to resources from their previous affiliation.

#### 5.3.3. Do: Communicate with service providers to inform them of timelines for deactivating identities.

When a user is scheduled for deactivation, the results of that can be far reaching. Not only might they lose access, but accounts will be deprovisioned from services, data deleted, and group memberships terminated. Whether you're doing this for a one-off staff member who has left or the entire graduating class of last semester, service owners should be informed ahead of time. In general, it's enough to make sure that service owners know your schedule: staff are deactivated after this many days, students after this many, *etc*. However, in cases where you're going to deactivate a large batch such as the example of the graduating class, an extra communication to service providers to remind them of the specific upcoming event can help prevent a lot of potential problems.

#### 5.3.4. Don’t: Deactivate or delete identities without communicating.

It may seem obvious, but it gets overlooked far too often. Before you terminate a user's access because of an affiliation state change, let them know. Provide information on things they might want to do such as graduated students setting up email forwarding or staff members downloading any relevant content that they're allowed to take with them upon their departure from the organization. Communicating ahead of time can also help to prevent mistaken deactivation such as a student who you believe has left but was just late registering for classes.

# 6. Passwords, Multi-Factor Authentication, and Provisioning

Login credentials, such as passwords, "passwordless" hardware tokens, cell phone "second factor" apps, and combinations of the preceding (*i.e.*, multi-factor authorization), when coupled with a user's identifier, are the security mechanisms protecting that user from the risk of being impersonated by someone else. Here is some advice to help you deliver secure authentication, appropriately balanced against end-user pain and frustration.

## 6.1. Password Rules and Policies

#### 6.1.1. Do: Limit the number of different passwords that users need to remember.

It’s hard enough these days to keep track of the many passwords we have in our personal lives. Adding multiple passwords for work or school will inevitably lead to forgotten passwords, user frustration, and help desk tickets. Use a central password store that all services can authenticate. Have a single password that grants access to everything that a user accesses. Where that’s not possible, consider synchronizing passwords between password stores.

#### 6.1.2. Do: Encourage single sign-on.

The only thing better than a single password is single sign-on. Rather than having to log into each service using the same password, architect your services such that a user only needs to sign in once in a given session. In a browser, things like SAML-based authentication against an identity provider that supports a persistent session can make this easy to set up and add security to your environment. On the desktop, setting up things to leverage and trust the user’s desktop authentication can accomplish the same result. In addition to user convenience, this adds security as the user won’t have to type their password into lots of different sites and services.

#### 6.1.3. Do: Consider the password policy advice from NIST (currently Special Publication 800-63B).

NIST has done extensive research and engaged numerous experts in developing these materials. There is good advice here.

#### 6.1.4. Don’t: Require frequent password changes.

It used to be a good, security-conscious idea to require users to change their passwords on a regular basis. With the rise of multi-factor authentication, this has become much less important. With multiple factors required to access a resource, a hacker can’t do much of anything if they crack or steal the password. Furthermore, not requiring users to change their passwords on a regular basis avoids forgotten passwords which generate help desk calls and doesn’t encourage a user to write down their password on that sticky note stuck to their monitor.

#### 6.1.5. Consider: Passwordless authentication.

"Passwordless" authentication tokens, usually based on FIDO2 protocols, may be a good fit in environments where having possession of a physical token may be more appropriate for authentication than knowledge of a secret password. (See also "Assignment of additional authentication factors" below.)

## 6.2. Initial Password Setting

#### 6.2.1. Do: Transmit account claiming information securely using activation codes or short-lived links.

Care should be taken the first time a user sets a password. The best approach is to provide some temporary, one-time-use item to the user to grant them access when initially setting the password. This can be a link that expires after a short amount of time or a random string of characters making up an activation code that's sent to them through some secure out-of-band means.

### 6.2.2. Do: Perform additional identity proofing during the account claiming process.

While a secure link or access code is a good start toward secure setting of initial credentials, it's only the first step. Just like multi-factor authentication uses items of different types such as something you have and something you know, first-time setup of credentials should do the same. In addition to the secure link, ask the user to answer questions that a hacker wouldn't know the answers to.

## 6.3. Assignment of Additional Authentication Factors

### 6.3.1. Do: Use multiple factors for authentication when possible.

Multi-factor authentication is becoming commonplace. It can certainly make things more secure when you require a user to present more than a secret string in the form of a password to gain access. It also alleviates the need for highly-complex passwords, a benefit for end-users. There are many technologies built to open standards for multi-factor authentication, including FIDO2, as well as older standards, such as HOTP and TOTP. There are also commercial authenticator apps, such as Google Authenticator and Duo. Shop around, and choose the ones right for your institution in consideration of accessibility, the availability of software support in browsers and online services, avoidance of vendor lock-in, and support for service providers that require specific MFA technologies.

### 6.3.2. Do: Use additional validation to password for adding or modifying MFA.

If a user only needs a password to gain access to the MFA configuration, then you might as well not have multi-factor authentication at all. A hacker who got a user's password could log into the user's MFA settings and change those settings to something the hacker could leverage instead. Make sure that, to change MFA settings, a user must perform an MFA authentication or some other backup method to verify that it's really them.

## 6.4. Provisioning and Deprovisioning of Credentials

Provisioning credentials into a service provider may be the only way to provide at least an approximation of a single sign-on experience for your users when the service provider cannot be federated (*i.e.*, must do its own authentication). This is, however, a security risk. It increases your risk of unauthorized exposure of the credentials, perhaps outside the scope of your direct control, if the service provider is not operated by you.

### 6.4.1. Don't (if possible): Provision credentials when a federation option is available.

Syncing passwords with service providers increases the complexity of password changes and resets, and increases risk associated with password exfiltration. Protocols such as SAML and OIDC are widely supported and give service operators no access to institutional credentials. Integrations with LDAP, AD, or Kerberos allow you to avoid syncing passwords, but may give service operators access to password cleartext, so SAML and OIDC should always be the preferred path.

### 6.4.2. Do: Maintain records of all service-specific usernames associated with each institutional identity

While federated authentication is much preferable to service-specific usernames (and credentials), a service-specific approach may be unavoidable, as observed above. It may even be necessary to create multiple usernames for a user that has been authorized for different roles on a single service (*e.g.*, staff, student, or service administrator). Assure that your IAM system maintains a record of all service-specific usernames that have been provisioned for each user on each service to support deprovisioning and changes to roles and other authorization-affecting attributes.

Note that there may be aspects of the service's internal account management that you may need to (or have the opportunity to) address in your IAM. This is particularly true if you are forced to share a single service-specific username among multiple people. You should, for example, consider a password vault, notifications to responsible individuals, and other processes to assure the service's security as people are added to and removed from the service.

### 6.4.3. Do: Avoid service-specific passwords or any password on the service side whenever possible.

Service specific passwords are confusing to users, and increase the complexity of password changes and resets. When using the SAML or OIDC protocols, ideally no password needs to be provisioned to the service. If the service still requires a password, even though it won't be used, then an appropriately long, randomly-generated complex password should be provisioned.

### 6.4.4. Do: Use federated authentication with a unique (pair-wise) identifier for each service provider.

Whenever possible use federated authentication to avoid the issues of password sharing. Unless multiple SPs need to share a common identifier (and policy allows such sharing), assign pair-wise identifiers to enhance privacy and to simplify transitions when identifiers must be changed.

#### 6.4.5. Do: Establish criteria to assure that the service provider's security measures for protecting credentials are comparable with yours.

Document these criteria and hold the service provider accountable. Remember that a breach of their system will be a breach of your IAM System and, potentially, all other service providers that rely on it. Your security people will be a good resource for developing the criteria.

#### 6.4.6. Consider: Periodic audits of the service provider's compliance with your criteria.

The need for this will depend on a number of factors. Consult with your security people for what is appropriate.

#### 6.4.7. Consider: Deactivating provisioned accounts, rather than deleting.

You may need to reactivate quickly, preserving the user's resources and state. On the other hand, there may be licensing fees that must be paid, even when an account has been deactivated. (See, also, the Deprovisioning section below.)

#### 6.4.8. Do: Keep enough information around to, at the minimum, prevent reissuing a username.

In addition to preventing reuse of a username, there may be files and other resources that may be needed in the future.

#### 6.4.9. Consider: In addition to the first factor, deactivate second-factor tokens if the account no longer has access to anything.

Multi-factor authentication is great until a user returns to your institution several years later with a new phone number, email address, or whatever external entity you were using for the second factor. Suddenly, the user can't get in to set up their account because a one-time password or similar code is being sent to a phone number that they haven't had in years. There's little to no point to leaving multi-factor methods active once a user no longer has access to anything of use. It only leads to challenges when they return, and has the additional possible side effect of increasing licensing costs if the user is occupying a license slot with an MFA vendor. As soon as the user has been deactivated and no longer has access to anything of use, deprovision second factors.

Also consider the issues in the Deprovisioning section, below.

# 7. Service Provisioning

"Just in time" provisioning is generally preferred over "just in case," as it minimizes data sharing, license costs, and system overhead if the provisioned account is never used. Nevertheless, many services require "just in case" provisioning. There may be solid business reasons why a service needs to know about a user before that user logs in (*e.g.*, to tell the user *how* to log in), that cannot be enabled by identity assertions received at the start of a session. It's also possible that a service is simply not technically capable of receiving new identity information at the start of a session. This section discusses the issues associated with service provisioning.

## 7.1. Reconciliation

#### 7.1.1. Do: Ensure that source and destination are in sync.

Deploy processes that ensure that synchronization errors, such as missed transactions, are corrected promptly.

#### 7.1.2. Do: Have both targeted and full reconciliation (fully match accounts).

Periodic full reconciliation for all information associated with all accounts is important to repair errors due to lost transactions. The need for targeted reconciliation (*e.g.*, for a particular user) can be used to repair errors or to enable (or disable) authorizations for a user, before the next full reconciliation is run.

## 7.2. State Changes and Fine-Grained Authorization (Continuous Access Evaluation Protocol)

#### 7.2.1. Do: Look at more than institutional data for fine-grained state changes.

Session data such as a mobile user's location also needs to be communicated (geofencing)

#### 7.2.2. Do: Keep up-to-date on emerging technologies in this area.

Fine-grained authorization is a developing field.

## 7.3. Communicating Updates to Service Providers

#### 7.3.1. Do: Have a reconciliation process to correct missed messages or reconcile out-of-sync changes in the case that the service provider did not respond to the change.

Incremental updates can, at times fail. A periodic reconciliation process will heal any damage.

### 7.3.2. Do: Deploy robust provisioning interfaces.

Ensure that your provisioning interfaces are robust, able to fail gracefully in the event of connection errors, system failures, *etc*. When failures do occur, take care to avoid corruption of data or other damage to either of the systems. Also ensure robust performance, particularly during large updates (*e.g.*, at the start of an academic session).

### 7.3.3. Do:  Have both incremental and full reconciliation.

Having near real time integration typically means detecting incremental changes in your identity registry or source system and pushing just the changed entries to your services and directories. Occasionally an incremental change may fail or be incomplete, or despite trying to prevent it, data might get updated in a service provider. Maintain a process that is capable of reconciling all of the entries in a service provider and verifying that the data matches your “source of truth” identity registry and source systems. When designing incremental change processes, consider the degree to which you can detect and correct inconsistencies without undue system overhead. For example, when applying an incremental change for a specific user, you might verify all provisioned data for that user, at least on services affected by the change.

### 7.3.4. Do: Make sure to delete service provider entries before deleting in the person registry.

If entries are ever removed from your identity registry, ensure you have a process that removes accounts in any service providers first, and the process assures the transaction succeeded. Orphan service provider entries are not only difficult to clean up, but can leave open application access that should have been removed.

### 7.3.5. Don’t: Permit updates from other sources to overwrite updates from the person registry.

It is important that account and authorization data carried by registries reflects the source of truth that is your identity registry and other authoritative data sources. Attributes should not be changed by other systems, or be updated directly in the service provider. In cases where the target directory must allow direct updates for business reasons, such as is frequently the case with Active Directory, isolate via OU structures the accounts that are managed by your identity system and prevent any other updates in those OUs.

## 7.4. Deprovisioning

Deprovisioning is the removal of access to provisioned services and plays an important role in maintaining the security of electronic systems. A review of an individual’s access should occur anytime that individual’s affiliation with the institution changes.  Common review times are when an employee terminates and when a student graduates but can also include times when an individual’s role within the institution changes, such as an employee moving to a new department or a student switching majors. A best practice with respect to deprovisioning is to remove the entitlements and authorizations to services rather than just disabling the authentication credential. This decreases the potential for someone to inherit inappropriate access as would be the case if an institution recycles usernames and the new “johndoe” user automatically has access to all the services that the old “johndoe” had because the entitlements were never removed. It’s important to have clearly documented and published rules and time frames regarding deprovisioning. Under what circumstances will I lose eligibility to Service X? How long will it be after I graduate or leave employment before Service X goes away? A best practice would be to send “pending service expiration” notifications or reminders to an individual during that grace period time frame. Not only will your users thank you but it could prevent the extra work of reinstating access if a mistake was made regarding eligibility or the user’s eligibility changes during the grace period time frame.

Sometimes deprovisioning can also involve removal/disabling  the accounts  on the downstream/target service . For example, terminating an employee can result in removing their Box account (provided you have a policy for this).

You should consider the flexibility of supporting both deprovisioning via removal of access and also removal of accounts for the target services.  Make sure the business rules and account mapping support both configurations.

### 7.4.1. Do: Gather all data you need for reporting, auditing, reactivating, *etc*. before deprovisioning.

There are many reasons why you may need to access information about deprovisioned access to services. The information may also be useful when it's necessary to verify a user's past ownership of the identity, as well as for forensic investigation. Make sure you have everything you may need before deleting.

### 7.4.2. Don’t: Forget deprovisioning, even when using just-in-time provisioning.

Login events are typically used for just-in-time provisioning, but they cannot be used for deprovisioning, since you cannot expect a login event. Some other method, probably driven by periodic reconciliation processes will be needed when it's necessary to purge information stored by the service or, for example, to reclaim the service provider's end-user licenses or other resources.

### 7.4.3. Do: Deprovision authorizations.

It is sometimes the case that some, but not all, permissions must be removed for an individual's use of a service, based on changes received from identity sources or manual changes to group membership, *etc*. Full deletion is not always the correct action, particularly if other information is stored within the service, such as conversation transcripts, videos, files, *etc*.

#### 7.4.4. Do: Give user an opportunity to migrate data, tools, instructions, *etc*. before they lose access (depatriation).

When appropriate, be sure end-users are given sufficient warning to allow them to obtain a copy of their data from a service for which deprovisioning is pending.

#### 7.4.5. Consider: Whether you need a process for transferring ownership of the user's data to their unit/manager/*etc*.

The "user's" data is often really the institution's. When this is the case, you will need a process for transferring that data to a successor.

#### 7.4.6. Do: Plan for potential repatriation (in or out: claiming accounts created before provisioning, or extracting data at departure/deprovisioning).

Deprovisioning is often followed in short order with a request for reprovisioning. Be prepared for this..

#### 7.4.7. Do: Set up processes ahead of time and provide users with an opportunity to preserve data (tools, instructions, *etc*) before deprovisioning occurs.

#### 7.4.8. Do: Consider that accounts may exist before provisioning occurs and plan for dealing with it.

This may or may not include permissions from previous eligibility that should have been cleaned up at deprovisioning.

#### 7.4.9. Don't: Accidentally restore old permissions that should have been cleaned.

Repatriation may not include all of the old permissions. Restore only what is needed at the current time.

#### 7.4.10. Don't: Leave deprovisioning for later when deploying a new service.

When deploying a new service, the emphasis is getting users up and running. This tilts immediate priorities to provisioning, rather than deprovisioning. Also, once the service is in production use, there are always other "immediate" priorities than can significantly delay setting up deprovisioning processes. This is a common phenomenon, but there are potentially serious security risks in not having effective deprovisioning processes, as well as higher costs for licenses that no longer in use and unplanned (often urgent) demands for personnel time to perform manual account deactivation. Always strive to deliver provisioning *and* deprovisioning at the same time.

### 7.5. Considerations for Cloud Services

Cloud services, particularly commercial cloud services, present unique issues for provisioning and deprovisioning, due to likely use of proprietary technical solutions and difficulties with navigating corporate structures to find the "right" people. For more information see the Provisioning and De-provisioning section of the Cloud Services Cookbook.

## 8. Target Directory Provisioning

Directories are an interesting form of service provider, in that they typically are used as identity sources for other service providers. This has implications for how you provision those directories.

### 8.1. Linking Identities between Directories

#### 8.1.1. Do: Have one or more attributes that are unique and immutable to link identities between source and target.

It is important to be able to resolve accounts in various systems that belong to the same identity. Having one or more attributes that are in your identity registry, and in each of the target directories will allow you to quickly identify accounts as changes in role, activation, attributes, and entitlements change.

#### 8.1.2. Do: Create an opaque institutional identifier used solely for linking.

Maintain an identifier to facilitate linking your directories and identity registry that doesn't contain strings that are meaningful to other aspects of the users identity. Strings like name based usernames, role based account naming conventions will change over time. The more opaque and limited the scope of linking identifier is, the less likely it is to change and break the association of a person across directories and registries.

#### 8.1.3. Don't: Use things like NetID that may seem immutable now.

NetIDs are tempting identifiers since they are often recognizable, published, and known by the user, however they are generally not a good choice for linking. Often NetIDs are name based or user chosen, and as with any non-opaque identifier are subject to change. Also, certain targets will have requirements that add scoping strings to the NetID for use as an authenticator, for example User Principal Name in Azure or Primary Email in Google. Not having a simple identifying attribute that is exactly the same in your target directories and your registry my hamper identity resolution.

#### 8.1.4. Do: Have a unique identifier for each of the target directories and make this mapping available to the provisioning process.

This will be useful to differentiate between target directories that may share, for example, the same Relative Distinguished Name (RDN), such as LDAP and AD.

### 8.2. Communicating Updates to Target Directories

#### 8.2.1. Do: Use a robust process to push changes in as close to real-time as possible.

Target directories are frequently the primary source for applications to learn about new identities and retrieve needed attributes and role information about the identity holder. Keeping the data in target directories as close to real time as possible means users gain, and just as importantly lose, access to applications that use directories for authentication and/or authorization. In the event of large updates (e.g., at the start of an academic session), ensure that appropriate performance is maintained for the service providers that rely on the directory.

## 9. Authorization

Authorization is an extension of institutions' processes for delegation of authority into digital services and resources. Institutional policies determine which people are allowed to access a service or resource, and what those people are allowed to do. Those policies may grant access entirely on the basis of institutional person data already known to your IAM system, such as affiliation, major, department, *etc*. (often referred to as "business" or "functional" roles). Alternatively, the policies might require an explicit decision by someone responsible for the service or resource.

A fundamental principle in IAM practice is "AuthN  AuthZ". In other words, a user's ability to authenticate (AuthN) should not imply authorization (AuthZ). Authentication only identifies who the current user is, not what that user is allowed to do (authorization). Authorization is determined based on information about the current user that is contained in the IAM system. That said, it should be noted that protocols such as SAML and OIDC appear to combine both authentication and authorization into a single step by identifying the current user and transmitting information about that user in a single transaction. Logically, though, the SAML IdP must perform authentication before gathering information to support an authorization decision, then transmit the results of both operations in the same transaction.

### 9.1. Types of Roles

**Business Roles** are generally used to describe high level affiliations or duties/functions within an organization.  Examples are : Employee, Staff, Faculty, Staff, Student, Alumni, Admit Coming, Guest, Contractor, *etc*. These can be considered high-level or “course grained” roles, as they may only tell part of the story of an entity, and may be used for basic entitlements like software licensing or building access. Business Roles are usually determined by institutional person data obtained from the authoritative source systems.

**IT Roles** are generally used to assign access rights and entitlements for specific services and, therefore, are more fine grained. IT Roles can be further categorized as follows:

- **Application Roles** describe end users' functions and entitlements, and can be further divided into:
    - Admin Roles, such as “IdM admin tool” or ”Box Admin User”
    - End-User Roles, such as ,”Office 365 User” or "Box User”
- **Asset Roles:** This describes assets  and devices that can be provided to end-users, such as hardware tokens, PC, Laptop , ID badge, mobile, *etc*.

Oracle's Using Role Types to Design Flexible Roles provides more advice for the structuring of roles.

In practice, roles are often implemented as **Groups** (collections of users), combined with any additional information needed to describe the associated access rights and entitlements.

### 9.2. Methods of Authorization

The most used methods for determining access rights are Role Based Access Control (RBAC) and Attribute Based Access Control (ABAC). There is a growing body of literature in this area. Wikipedia provides good starting points:

- Role-based access control
- Attribute-based access control

(When reading these references, note that the authors often assume a person has one role. In academia people often have multiple roles, such as multiple job appointments, or being both student and employee.)

You build a collection of roles (what a person does for or with the institution), possibly add more personal or environmental data (attributes), and authorize accordingly. For example, a student (role) in engineering (detail), currently on campus (environmental) is authorized to access the School of Engineering's student portal.

Whenever possible, authorization should be determined by Business Roles and identity Attributes that are determined through automated processes. This allows authorization and de-authorization to be automated, making sure that exactly the right people have access. If "every Registrar's office employee should have access to student records", or "every engineering student should have access to this file share", authorization can be granted and revoked as a person's job or study field changes. When authorization is determined manually, de-authorization tends to lag behind, leaving people with access they should not have.

All authorization processes should have periodic re-examination, often called "attestation" or certification. If you have an automated process for "every Registrar's office employee should have access to student records", verify that the rule is still correct, as well as the logic for determining a Registrar's office employee. If authorization is manual, then each person who has access should be manually verified. (See Assuring Provisioned Authorization Is Correct, below, for more information.)

Automated creation of roles from institutional data is good, but you will need to create roles that cannot be determined in this fashion. Not every access can be determined from institutional data. Also, the effort to automate may be greater than the benefits. Start automating where you can get the most benefit and reduce risk.

Authorization is usually implemented in one of two ways. Authorization decisions can be made in advance and stored within the application (periodic updating as changes needed) that will execute them, or the application can call an "AuthZ service" to request an authorization decision at the time of need. The former is the traditional and still most common method, but the latter has advantages and is growing in use. You will want to support both methods.

Commercial systems are available for implementing authorization. When investigating commercial systems, make sure that they can handle the complexity of your affiliations -- people who are both employee and student, multiple job appointments, *etc*. There is also a community-supported system, InCommon Grouper.

## 9.3. Designing Roles, Attributes, and Groups for Authorization

Authorization decisions can be made on the basis of users' *roles*, *i.e.*, what they do for or with the institution; this is called Role Based Access Control (RBAC). What a user does for/with the institution implies a set of responsibilities and authorities, and those authorities imply access to services and what those services will do for the user. *Groups*, lists of users, are used to represent the users that fill each role. As such, RBAC authorization decisions are relatively stable over time.

Other *attributes* can also be used to make authorization decisions; this is called Attribute Based Access Control (ABAC). Examples of such attributes include identity information, such as academic department, but may also include attributes such as time of day that are difficult to express with roles when criteria are relatively dynamic over time. Judicious use of RBAC and ABAC will result in a simple, yet robust design.

It should be noted that the distinction between the terms "role" and "group" can be fuzzy. Many computer systems provide access to a resource by making everyone who is authorized a member of what they call a "group" or "security group". Some systems provide access similarly with "security roles". This is so common that the terms "groups", "roles" and also "membership" may be used to describe authorization processes, regardless of the mechanisms actually used in a target system. In this document, we will try to be consistent with the definitions provided above.

Whenever possible, business roles and groups should be composed automatically from institutional data. Think of a person's data as determining group membership, and then granting access to "all members of this group".

At the highest (coarsest) level, roles describe a basic function in the institution-- faculty, student, guest, contractor, employee. Depending on your needs, you may want to split employee into regular and temporary, or add a campus for a large system (*e.g.*, FacultyMadison). At a university, it is common for people to have more than one affiliation. You can use these roles to drive eduPersonEntitlement, and to determine access to campus level services (*e.g.*, all students get Google mail).

The next finer level will be determined by the access needs of the organization. If you need to grant access based on school or college, then define groups such as 'student in engineering' or 'faculty in public health'. Here too, people may be in multiple groups.

At successively finer levels, you may need to define groups such as 'student in English 123' or 'senior accountant in engineering'. These groups would enable you to automate access grants to course materials or the financial system.

The following are considerations for the design of your roles, groups, and attributes.

### 9.3.1. Do: Start simple.

The highest level roles and groups will allow you to determine access to campus-wide services. Their large size will help you work out mechanisms for computing, storing, and delivering authorizations.

### 9.3.2. Do: Have a clear naming convention.

This includes components to identify who owns the group, who or what the group represents, what resource the group has access to, and how that group has an effect on that resource. Naming varies based on the type of registry that is storing the data. For example, Grouper uses its folder (stem) path as part of the group name: "app:splunk:index:firewall2-read" shows the group "firewall2-read" as part of the stem "index", beneath the "splunk" stem, and thus under "app". This clearly indicates the ownership (Splunk service), the resource (some index named firewall2) and the effect or action (read).

### 9.3.3. Do: Express RBAC for Business roles.

These permissions usually map to specific operations on one or more resources. The idea of RBAC is that you map the role once to the specific set of permissions. The on-going maintenance is simply the membership of those entities in that RBAC role.

### 9.3.4. Do: Express ABAC for your IT roles.

Attribute Based Access Control can give you more flexibility to add additional information to the role. For example, A Business role “Staff” can be assigned multiple IT roles as attributes such as “Box Admin” “Office 365 User”. Similarly, a person with a location of "Communicable Diseases Lab" might be granted access to an application that admits guests to the facility. This will help you avoid role explosion. Reference NIST 800-162 for more details. https://csrc.nist.gov/publications/detail/sp/800-162/final

ABAC can facilitate complex rule sets from multiple identity attributes and allow for the creation of automatic groups based on those attribute values. For example, any student with two attributes “HomeCollege=Engineering” and “CurrentTermRegistered=true” could be the supporting details going into an ABAC group of all currently-enrolled engineering students.

### 9.3.5. Do: Consider the tradeoff of adding, for example, location or department as roles, as opposed to attributes.

Since a role maps to permissions, generally an organizational component (academic unit or building) does not map to a permission. For example, “The Math Building” doesn’t convey a permission, but rather a resource. You can create cohorts or groups based on a person’s location or department, which can be used in ABAC policies.

### 9.3.6. Do: Follow common concepts provided in documentation such as InCommon’s Grouper Deployment Guide.

The Grouper Deployment Guide's advice is largely independent of your particular group/role management platform.

### 9.3.7. Do: Have clear and complete documentation of the design of your authorization architecture.

Your clients, co-workers, successors (and auditors) will love you for this.

## 9.4. Implementing Roles, Attributes, and Groups for Authorization

Recognizing that authorization is an extension of your institution's policies for delegation of authority, give careful thought to the processes for assigning and removing administrative privileges that allow a user to grant access to other users or manage access control policies.The following are considerations for the implementation of your roles, groups, and attributes.

### 9.4.1. Do: Consider how you will handle authorizations that change e*n masse* with an academic term or session.

This is especially true if you authorize access based on course enrollment. You need to ensure appropriate performance while processing large numbers of updates.

### 9.4.2. Do Consider allowing grace periods for de-authorization.

Consider grace periods to ensure continuity of service (and to reduce the disruption of large volumes of changes smoothly) when de-authorization is likely to be followed by re-authorization in a short period of time. For example, authorizations granted to all students are probably best continued between the end of one term and the start of the next, unless it is known that a student is not returning. Also, consider how you will inform people that they have entered a grace period.

### 9.4.3. Consider: How to grant access to someone who does not meet the automatic criteria, or deny access to some who does meet them.

Consider how can you do that in such a way that the exception will have time limits, or trigger periodic reevaluations.

### 9.4.4. Do: Consult with your institution’s auditors on any requirements for maintaining a history of access decisions.

You will likely find your internal auditors to be strong partners as you build your IAM service.

### 9.4.5. Do: Keep authentication systems separate from enforcing authorization decisions, where possible.

Attributes and entitlements should be handed off to the service for it to handle its own authorization enforcement. For Services that lack proper authorization mechanisms, use a groups registry (*e.g.*, Grouper or groups in Active directory) to set an access policy where the IdP becomes the policy enforcement point by allowing/denying access at the IdP.

### 9.4.6. Do: Adopt product-specific best practices where applicable.

A good example is the Grouper Deployment Guide for Grouper. We emphasize this model, as Grouper has grown to become a popular open-source solution to group and role management among higher-ed institutions. As always, avoid adopting practices that create heavy dependencies on your current platform.

## 9.5. Computing, Storing, and Delivering Authorization Decisions

Authorization decisions may be pre-computed and stored as roles, groups, and/or attributes, or they can be computed "just in time," when a service or resource requests it. The choice depends on a number of factors, most importantly whether the decision includes consideration of information that changes over time, like time of day or location, or if the decision includes only information that is updated at known times by the IAM system. Other factors, such as storage vs. compute cost and complexity may also be relevant.

### 9.5.1. Do: Reconcile and re-compute the decisions periodically when authorization decisions are pre-computed.

Consider the acceptable lag time between when a person's information is updated and when an authorization decision is computed and stored. This is often very short (*e.g.*, sub-second), requiring event-driven computation, but even when it is not, regularly-scheduled reconciliations should be established to meet institutional and end-user expectations, as well as to recover from missed event-driven transactions.

### 9.5.2. Do: Implement "just in time" computation of authorization decisions when policy dictates consideration of factors that change over time.

This is information, usually associated with the current session, such as time of day, location, authentication method (*e.g.*, password, X.509 certificate, biometrics, MFA token) or behavior patterns.

### 9.5.3. Do: Make sure the service provider is authorized to inquire about the particular authorization.

Ensure that your attribute release policies support service providers' need to query for roles and attributes that have been determined to be required for authorization decisions.

### 9.5.4. Do: Communicate fine-grained institutional changes to services when appropriate.

Ensure that service providers have the information that it's been determined they need.

# 10. Assuring Provisioned Authorization Is Correct

Your IAM system likely interfaces with multiple sources of thousands to hundreds of thousands of identities to support tens to hundreds of services and resources. The advice in this cookbook will help you deal with this large scale, but errors will undoubtedly occur. This section discusses attestation and audit, two IT governance processes designed to catch those errors and correct them.

## 10.1. Attestation to Review Access Decisions

Attestation is a process whereby the people who are responsible for services are asked to verify that the IAM system is correctly configured to support authorization decisions by their service. This includes both how users are selected to be provisioned, as well as any entitlements that are associated with those users. The attestation process should be designed to verify entitlements, roles, and other attributes that are used to provision and deprovision service-specific permissions, as described in the Service Provisioning section.

### 10.1.1. Do: Focus on what really needs a human to consider.

If access is granted by rule based on institutional data, only the rule needs periodic attestation -- not the individual grants.

### 10.1.2. Do: Make attestations understandable, so it is clear what is being attested to.

Make sure this is well documented. If possible create the documentation in partnership with the person responsible for the service at the time the service is integrated with your IAM system.

### 10.1.3. Consider: Attestations' effect on resulting deprovisioing processes.

Changes to a person's entitlements, roles, *etc*., due to attestation, may initiate provisioning and/or deprovisioning. This should be considered when establishing grace periods, *etc*.

### 10.1.4. Consider: Attestations that are not completed should result in access suspension until completed.

If the party responsible for the attestation has not responded to the attestation review on time, before the system can take action to remove access, it should first send escalation a few times before access deprovisioning happens. The escalation should support multiple levels, both horizontal (peers) and vertical (management).

## 10.2. Audit

Periodic audits of IAM processes is a more holistic review of your IAM system and its relationships with the rest of your institution's IT ecosystem. It may include, for example, service managers' policies for the assignment of roles and entitlements, as well as the operation of your IAM system, in light of institutional policy, regulatory requirements, and industry best practice.

### 10.2.1. Do: Involve your institution’s auditors early and often.

They can be a great help in designing your system and processes to facilitate regular reviews to minimize the time incorrect information remains in your system.

### 10.2.2. Do: Schedule regular “full review” processes starting from destination content and working backwards.

Audits require a reasonable amount of effort, easily postponed. Establishing a a regular schedule for this work will help ensure that it is not forgotten.

### 10.2.3. Do: Schedule regular review of authorization processes.

Because auditing data can add up so quickly, consider adding/exporting data to an external database on a schedule compatible with the regular "full" and authorization reviews. This will help performance significantly .

# 11. Product Lifecycle

Over time, an increasing number of services will rely on the provisioning functions of your IAM System. When selecting provisioning products (software or service, open source or commercial, on-premise or off-premise, SaaS or IaaS) for use in your IAM System, as well as the services you are provisioning, it is important to consider the eventual end of life of that product, as well as how you will integrate that product initially. In addition to the considerations provided here, a tool is available in a Google Drive folder at https://goo.by/cvr77, containing a spreadsheet (*Master Revised BTAA IDM Product Evaluation Metrics*) to help you evaluate the products you are considering, as well as a subfolder (*Community Submissions*) where, if you wish, you can share your findings with the community. Instructions are provided in the spreadsheet.

## 11.1. Product Onboarding

There are many issues you must address in the selection and deployment of a new product. The following are specific to provisioning.

### 11.1.1. Do: Have a plan for loading your data into the product.

This is likely to be a combination of migrating existing provisioning-related data, as well as new data that may be required by the new product. Services you provision may also store user-owned data that must be loaded, particularly when migrating from a competing product.

### 11.1.2. Do: Understand (and mitigate) the risks for sensitive data.

This is particularly true of cloud-based products.

### 11.1.3. Do: Understand the provisioning "hooks" in the product.

Understand what out-of-the-box connectors included in the products. Can the product be customized to add or extend new connectors? This includes triggers to initiate provisioning actions, service provider integrations, *etc*.

### 11.1.4. Do: Understand how you will map roles and attributes to service provider permissions.

Maintain interface mapping documentation while onboarding service providers, keeping tabs on the attributes, default values, transformations, schema validations (field type, character lengths, nullable, *etc*). Also consider how attributes affect provisioned permissions. Can the mapping be configured (*e.g.*, via a GUI or by editing files) without developing software code?

### 11.1.5. Consider: How the product's license terms may affect cost as your institution's use of the product grows.

Is it a flat fee? Does it increase with population growth, number of "seats," storage requirements, *etc*.? Does it consider alumni, or other internal or external users? Does it use a standard size-ranking structure, such as IPEDS? Will the cost structure require you to restrict usage or to offboard no-longer-eligible users with shortened grace periods?

#### 11.1.6. Consider: Whether and how the product can be customized, and who can do the customization.

IAM system requirements are often very unique to each institution, requiring unique customization. This issue is particularly important for SaaS IAM services, where the vendor has control over the deployment, as well as the software implementation.

#### 11.1.7. Do: Consider all other technology acquisition issues that are not specifically related to IAM.

This includes ongoing maintenance, vendor stability, ease of deployment, *etc*.

### 11.2. Product Offboarding

When selecting and onboarding a new product, consider your eventual exit strategy.

#### 11.2.1. Do: Know how you will retrieve your data, workflows, procedures, *etc*. from the old product.

Avoid products that do not have clear ways to do this.

#### 11.2.2. Do: Know how sensitive data will be destroyed.

This is particularly important for cloud-based provisioning services.