

# User Provisioning Best Practices



# Contents


- 1 Introduction 1**
- 2 Terminology and Concepts 2**
  - 2.1 What is Identity Management? . . . . . 2
  - 2.2 What is Enterprise Identity Management? . . . . . 2
  - 2.3 What is Entitlement Management? . . . . . 3
- 3 User Lifecycle: Business Challenges 5**
- 4 Administration Within Application Silos 7**
- 5 Overview of User Provisioning 8**
- 6 Human Factors 10**
- 7 Enforcing Standards 11**
  - 7.1 Best Practices . . . . . 11
    - 7.1.1 Assigning unique identifiers . . . . . 11
    - 7.1.2 Object Placement . . . . . 12
    - 7.1.3 Security Entitlements . . . . . 13
    - 7.1.4 Change Authorization . . . . . 13
- 8 User and Entitlement Management Processes 16**
  - 8.1 Identity synchronization . . . . . 16
    - 8.1.1 When to use . . . . . 16
    - 8.1.2 Scope . . . . . 16
    - 8.1.3 How to use . . . . . 16
    - 8.1.4 Pitfalls to avoid . . . . . 16
  - 8.2 Auto-provisioning and automatic deactivation . . . . . 18
    - 8.2.1 When to use . . . . . 18
    - 8.2.2 Scope . . . . . 18
    - 8.2.3 How to use . . . . . 19
    - 8.2.4 Pitfalls to avoid . . . . . 19
  - 8.3 Self-service requests and delegated administration . . . . . 20

8.3.1	When to use . . . . .	20
8.3.2	Scope . . . . .	20
8.3.3	How to use . . . . .	20
8.3.4	Pitfalls to avoid . . . . .	21
8.4	Authorization workflow . . . . .	22
8.4.1	When to use . . . . .	22
8.4.2	Scope . . . . .	22
8.4.3	How to use . . . . .	22
8.4.4	Pitfalls to avoid . . . . .	23
8.5	Consolidated reporting . . . . .	24
8.5.1	When to use . . . . .	24
8.5.2	Scope . . . . .	24
8.5.3	How to use . . . . .	24
8.5.4	Pitfalls to avoid . . . . .	24
<b>9</b>	<b>Internal Controls</b>	<b>25</b>
9.1	Using Roles to Grant Appropriate Entitlements . . . . .	26
9.2	Enforcing Segregation of Duties Policies . . . . .	27
9.3	Periodically Reviewing and Correcting Entitlements . . . . .	29
<b>10</b>	<b>Integrations with Systems and Applications</b>	<b>31</b>
<b>11</b>	<b>Summary</b>	<b>33</b>
	<b>APPENDICES</b>	<b>34</b>
<b>A</b>	<b>Identity Manager Overview</b>	<b>35</b>

## 1 Introduction

This document describes and justifies user provisioning best practices in medium to large organizations.

It is intended to offer reasoned guidance to IT decision makers when they set security policies and design processes to manage user identities and entitlements across multiple systems and applications.

Look for the  marks throughout this document to find best practices.

## 2 Terminology and Concepts

### 2.1 What is Identity Management?

Identity and access management refers to a set of technologies and processes used to coherently manage information about users in an organization, despite the fact that identity data may be scattered across organizational, geographical and application boundaries.

Identity and access management addresses a basic business problem: information about the identity of employees, contractors, customers, partners and vendors along with how those users authenticate and what they can access is distributed among too many systems and is consequently difficult to manage.

### 2.2 What is Enterprise Identity Management?

*Enterprise Identity and Access Management (IAM)* is defined as a set of processes and technologies to effectively and consistently manage modest numbers of users and entitlements across multiple systems. In this definition, there are typically significantly fewer than a million users, but users typically have access to multiple systems and applications.

Typical enterprise identity and access management scenarios include:

- Password synchronization and self-service password reset.
- User provisioning, including identity synchronization, auto-provisioning and automatic access deactivation, self-service security requests, approvals workflow and consolidated reporting.
- Enterprise single sign-on – automatically filling login prompts on client applications.
- Web single sign-on – consolidating authentication and authorization processes across multiple web applications.

Enterprise IAM presents different challenges than identity and access management in Extranet (B2C or B2B) scenarios:

Characteristic	Enterprise IAM (typical)	Extranet IAM (typical)
Number of users	under 1 million	over 1 million
Number of systems and directories	2 – 10,000	1 – 2
Users defined before IAM system is deployed	Thousands	Frequently only new users
Login ID reconciliation	Existing accounts may have different IDs on different systems.	Single, consistent ID per user.
Data quality	Orphan and dormant accounts are common. Data inconsistencies between systems.	Single or few objects per user. Consistent data. Dormant accounts often a problem.
User diversity	Many users have unique requirements.	Users fit into just a few categories.

In short, Enterprise IAM has fewer but more complex users. Extranet IAM has more users and higher transaction rates, but less complexity.

## 2.3 What is Entitlement Management?

The Burton Group defines an entitlement as:

*An entitlement is the object in a system's security model that can be granted or associated to a user account to enable that account to perform (or in some cases prevent the performance of) some set of actions in that system. It was commonly accepted that this definition of entitlement referred to the highest-order grantable object in a system's security model, such as an Active Directory group membership or SAP role and not lower-order objects such as single-file permission setting.*

Definition by Ian Glazer, in *Access Certification and Entitlement Management v1*, September 9, 2009.

<http://www.gartner.com/technology/research.jsp> (login required)

Entitlement management refers to a set of technologies and processes used to coherently manage security rights across an organization. The objectives are to reduce the cost of administration, to improve service and to ensure that users get exactly the security rights they need.

These objectives are attained by creating a set of robust, consistent processes to grant and revoke entitlements across multiple systems and applications:

1. Create and regularly update a consolidated database of entitlements.
2. Define roles, so that entitlements can be assigned to users in sets that are easier for business users to understand.

3. Enable self-service requests and approvals, so that decisions about entitlements can be made by business users with contextual knowledge, rather than by IT staff.
4. Synchronize entitlements between systems, where appropriate.
5. Periodically invite business stake-holders to review entitlements and roles assigned to users and identify no-longer-appropriate ones for further examination and removal.

### 3 User Lifecycle: Business Challenges

As organizations deploy an ever wider array of IT infrastructure, managing that infrastructure and in particular managing users, their identity profiles and their security privileges on those systems becomes increasingly challenging.

Figure 1 illustrates some of the challenges faced by organizations that must manage many users across many systems.



Figure 1: User Lifecycle Management Challenges

In the figure, there are business challenges at each phase of the user lifecycle:

#### 1. Onboarding new users:

(a) **Delays and productivity:**

New users need to get productive quickly. Any delays in setting up access rights for new users cost money, in terms of lost productivity.

(b) **Requests and approvals:**

IT workers need to be certain that newly created accounts are appropriate. This usually means a paper process for requesting, reviewing and approving security changes, such as the creation of new accounts. This approval process may be hard to use, may require excessive effort on the parts of both requesters and authorizers and may introduce delays.

(c) **Redundant administration:**

Users typically require access rights that span multiple systems. A new user may need a network login, an e-mail mailbox, firewall access and login rights to multiple applications. These accounts are typically created by different administrators, using different tools. This duplication is expensive and time consuming.

## 2. Managing change:

Users often change roles and responsibilities within an organization. They may also change identity attributes (e.g., changes to a user's surname, contact information, department, manager, etc.). Such changes trigger IT work, to adjust user identity profiles and security rights.

Organizations face the same challenges in managing existing users that they face when creating new ones:

- (a) **Delays:**  
Reassigned users waste time waiting for IT to catch up with their requirements.
- (b) **Change requests:**  
Can be awkward to submit and may take time to approve.
- (c) **Redundant administration:**  
Similar changes are often required on different systems.

## 3. IT support:

In the context of routine use of systems, users often encounter problems that require technical support:

- (a) Forgotten passwords.
- (b) Intruder lockouts.
- (c) Access denied errors.

Collectively, these problems typically represent a large part of an IT help desk's call volume. This means both direct cost (support staff) and indirect cost (lost user productivity).

## 4. Termination:

All users leave eventually. When they do, reliable processes are needed to find and remove their security privileges. These processes must be:

- (a) **Reliable:**  
If an organization fails to deactivate the access rights of a departed user, then that user or an intruder impersonating him might abuse the infrastructure or compromise sensitive data.
- (b) **Timely:**  
Access termination must be prompt, to minimize the time window available for the aforementioned exploits.
- (c) **Complete:**  
It is not enough to deactivate a departed user's login IDs on major systems. Every access right should be revoked, to eliminate the possibility of abuse by users inside the network.

## 4 Administration Within Application Silos

Without an identity and access management system, users are managed by separate administrators, using separate software tools and often separate business processes, on each system and application. This is illustrated in Figure 2.

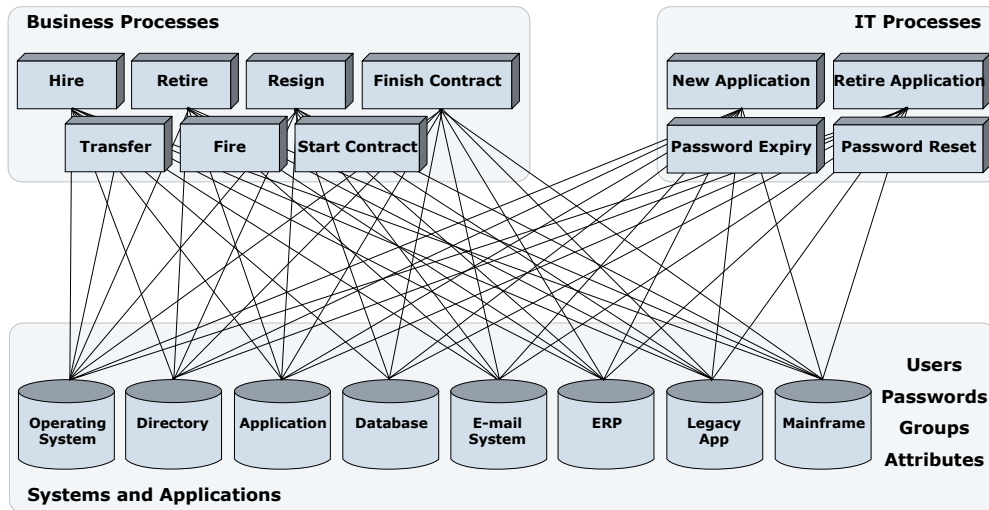


Figure 2: Managing Each Application in its own Silo

Identity and access management systems externalize the administration of user objects, replacing processes that are implemented within each system with new processes that apply uniformly to all users, across all applications. This simplified process is illustrated in Figure 3.

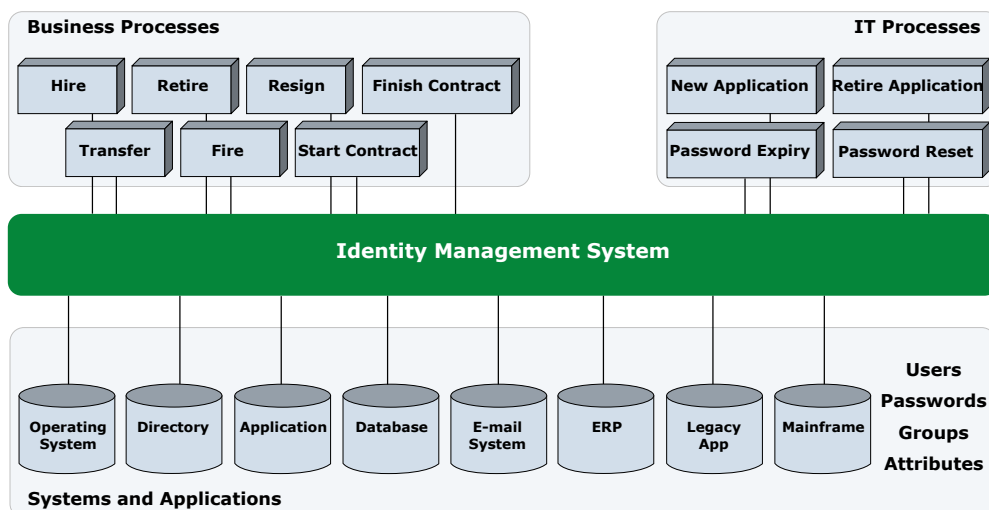


Figure 3: Externalizing the Management of Users and Entitlements

## 5 Overview of User Provisioning

A user provisioning system is shared IT infrastructure which is used to pull the management of users, identity attributes and security entitlements out of individual systems and applications, into a shared infrastructure.

User provisioning is intended to make the creation, management and deactivation of login IDs, home directories, mail folders, security entitlements and related items faster, cheaper and more reliable. This is done by automating and codifying business processes such as onboarding and termination and connecting these processes to multiple systems.

User provisioning systems work by automating one or more processes:

- **Auto-provisioning:**  
Detect new user records on a system of record (such as HR) and automatically provision those users with appropriate access on other systems and applications.
- **Auto-deactivation:**  
Detect deleted or deactivated users on an authoritative system and automatically deactivate those users on all other systems and applications.
- **Identity synchronization:**  
Detect changes to personal data, such as phone numbers or department codes, on one system and automatically make matching changes on other systems for the same user.
- **Self-service requests:**  
Enable users to update their own profiles (e.g., new home phone number) and to request new entitlements (e.g., access to an application or share).
- **Delegated administration:**  
Enable managers, application owners and other stake-holders to modify users and entitlements within their scope of authority.
- **Access certification:**  
Periodically invite managers and application owners to review lists of users and security entitlements within their scope of authority, flagging inappropriate entries for further review and removal.
- **Authorization workflow:**  
Validate all proposed changes, regardless of their origin and invite business stake-holders to approve them before they are applied to integrated systems and applications.
- **Consolidated reporting:**  
Provide data about what users have what entitlements, what accounts are dormant or orphaned, change history, etc. across multiple systems and applications.

As well, a user provisioning system must be able to connect these processes to systems and applications, using connectors that can:

- List existing accounts and groups.
- Create new and delete existing accounts.

- Read and write identity attributes associated with a user object.
- Read and set flags, such as “account enabled/disabled,” “account locked,” and “intruder lockout.”
- Change the login ID of an existing account (rename user).
- Read a user’s group memberships.
- Read a list of a group’s member users.
- Add an account to or remove an account from a group.
- Create, delete and set the attributes of a group.
- Move a user between directory organizational units (OUs).

## 6 Human Factors

Identity management is all about better administration of information about users: who they are and what they can access. Unsurprisingly, this often requires assistance from the users themselves to ensure their information is accurate and complete. Providing a user friendly system is essential to a successful deployment of the system.

Users need to be motivated to use the system, rather than reverting to older, manual processes. From a user's perspective, it must be easier, more obvious and more rewarding to use the automated provisioning system than to call the help desk.

Consider the impact of the system on users:

- If an enrollment process is required, for example to implement self-service login ID reconciliation.
- If a periodic review of security rights is implemented.
- If managers or application owners are asked to authorize security change requests.
- If security change requests must be using the system, which means that users must know where to find it and how to use it.

Where wide-spread user involvement is needed, special care must be taken to ensure success:

- A user awareness program should be considered, to ensure that all users understand what the system is, what it is intended to accomplish, where to find it and how to use it.
- User training may be required. Where large numbers of users are impacted and in general where there may be a significant time lapse between scheduled training and actual use of the system, it is preferable that this be computer-based training (CBT), embedded into the system, rather than in-person education, which is expensive and quickly forgotten.
- Users should be automatically reminded to use the system whenever their participation is called for.
- Allowance must be made for users who are too busy or simply not available to respond to the system – reminders, delegation of authority and escalation from one user to another.

It is sometimes also helpful to implement dis-incentives for inappropriate behaviour. For example, paper forms for access changes may still be accepted, but may be processed significantly more slowly than on-line requests.



## 7 Enforcing Standards

One of the weaknesses of manual user administration is that people are not consistent – they make mistakes. As a result, security administrators cannot be expected to reliably enforce standards regarding what access rights users should have.

An identity management system can and should enforce standards over how changes are requested, what they contain, how they are authorized and how they are fulfilled. This includes:

- Assigning unique identifiers:
  - Employee numbers.
  - Login IDs.
  - E-mail addresses.
- Object placement:
  - Placing new users in the correct directory container.
  - Creating mailboxes and home directories on appropriate servers.
- Default security setup:
  - Initial group memberships and role memberships.
  - Initial security ACL / permission setup on home directories, mail folders and desktop profiles.
  - Setting of security-related attributes on applications and directories.
- Change authorization:
  - Ensuring that change requests are submitted by authenticated users and signed if appropriate.
  - Ensuring that appropriate business stake-holders are invited to approve all change requests.
  - Deferring fulfillment until authorization is complete.
- Miscellaneous:
  - Default membership in mail distribution lists.
  - Disk quota allocation.
  - Tablespace allocation.

Care must be taken to define standard policy for each of the above items before deploying a user provisioning system, as described in the following sections.

### 7.1 Best Practices

#### 7.1.1 Assigning unique identifiers

Clearly, the actual policy for each type of identifier will vary between organizations. That said, some best practices that many organizations have found to be effective include:



- Assign users a short login ID and use that ID on all systems.
  - IDs should be short in order to be easy to type and compatible with all systems, including legacy.
  - This means a maximum length of 7 or 8 characters and use of only letters and digits in login IDs.
- Do support longer login IDs, such as full name, SMTP address or fully qualified directory path, but only as secondary identifiers. Longer IDs have their uses but they should not replace the short, unique IDs described above.
- Make login IDs globally unique. Do not assign the same ID to two different users on two different systems, or in two different OUs, as this can create problems with event correlation later.
- Never reuse IDs. Once an ID has been assigned to a user, it should represent only that user, in perpetuity. This protects the sanctity of audit records.
- Even where a hierarchical namespace is used, assign a globally unique ID to each user, typically in the directory CN. In other words, no CN should appear twice in a directory, in two different contexts. This helps when mapping IDs between systems with hierarchical and flat namespaces.
- Assign login IDs to people, not to positions in the organization. People change roles, but their login IDs should follow them, to support continuity of communication (e.g., same e-mail address) and accountability in relation to audit logs.

Many algorithms can be used to assign login IDs in compliance with the above guidelines. Examples include:

- Use the first three digits of the user's last name, followed by a 5 digit numeric sequence, to create a unique ID.
- Combine the first four letters of the user's surname, followed by first and second initial, followed by two characters to ensure uniqueness.

Ensuring global uniqueness and preventing reuse, means that a table must be maintained on some system to track all currently-in-use IDs, plus IDs that have been reserved but not yet created and IDs that were used in the past but are not currently active. Such a table is required to prevent ID reuse.

### 7.1.2 Object Placement

Placing new accounts in the correct directory container and creating their new mailboxes and home directory folders on appropriate servers and disk volumes is straightforward – it should derive directly from the directory's structure, the structure of the mail server infrastructure, etc.

A more subtle and difficult consideration is to track changes to user identity attributes and to automatically move accounts to different directory contexts, move mailboxes to new servers and relocate their home directories to reflect changes in a user's identity attributes.

For example, if a directory structure reflects an organization's departments and a user moves to a new department, then his account in the directory should be moved to the new department's OU.



Similarly, if the selection of an appropriate mail server to host a user's mailbox depends on the user's physical location and the user moves to a different branch office, then the user's mailbox should likewise be automatically moved.

### 7.1.3 Security Entitlements

In much the same way, a new user's default security entitlements should be based on standards for the user's location and job code. The key, as above, is to detect important changes to identity attributes and automatically make appropriate adjustments.

For example, if a user's initial group memberships were derived from the user's location and department and the user subsequently moved to a new location or different department., then some group memberships should be removed and others added, automatically.

In reality, this may be easier said than done, as it implies that roles have been defined for every valid combination of department and location and that a role change can be triggered. This may only be economical for combinations (roles) that are shared by many users.

### 7.1.4 Change Authorization

First, it should be clarified what is meant by a "change request:"

A change request is essentially a document, with several participants:

1. A single requester – human or automated.
2. A single recipient – a person or pseudo-person whose profile on one or more systems will be impacted.
3. Zero or more authorizers, who may be invited to review and approve or reject the request.

There may be other participants – alternate authorizers such as people to whom authority is escalated or delegated; workflow managers who monitor and respond to problems with the request queue; implementers, who manually fulfill some requests and more.

A request specifies one or more changes to the recipient's profile, which may include:

1. Create a new profile.
2. Deactivate or delete an existing profile.
3. Change some identity attributes, possibly including special cases such as the user's unique identifier(s) or directory container.
4. Create a new user object (login account) and add it to the profile.
5. Disable or delete a user object.
6. Add the recipient to a security group.

7. Remove the recipient from a security group.
8. Add the recipient to a role.
9. Remove the recipient from a role.

A request may be immediate – i.e., implement as soon as possible – or scheduled for some future date. It typically has a reason associated with it.

Change requests should be accepted and approved only if they are consistent with business requirements. This is typically done in two steps:

#### 1. Request validation:

Automatic inspection of a request to check whether it violates any business rules. For example, requests should not trigger violations of SoD rules, should not specify invalid department or location codes, etc.

#### 2. Request authorization:

Trivial requests, such as self-service updates to a user's phone number, can be processed immediately.

Requests that originate from a trusted system or person – for example, requests that are based on an authoritative data feed from a human resources system (HR feed) or that are entered by a very trustworthy person – for example, the CFO, may not require further authorization.

All other requests should be reviewed by business stake-holders before they are fulfilled.

For requests that do require authorization, the logical question is: who should approve them? There are only a few possible choices:

- **Managers** – typically of the recipient, but in some cases also of the requester.
- **Resource owners** – of resources that are impacted by the request. Note that resources may take many forms, including:
  1. Applications.
  2. Security groups.
  3. Roles.
  4. SoD rules (where the request includes an SoD violation).
- **Security officers** – for example, some organizations designate a key business user in each department or division to approve security changes for users within that part of the organization.

It is advisable to invite multiple types of authorizers to approve any given request. Typically this means inviting the owners of every resource being added plus the manager of the request's recipient.



Some exceptions to this rule inevitably come up. In particular, executives above a certain level in the organization may not require managerial approval for their change requests and due to their position, it would be pointless to ask for resource owner approval either (it would be granted by default).

Also, a user should not be asked to approve his own change requests – the answer will always be “yes.” This means that the workflow engine should check the list of authorizers prior to sending out invitations and if the requester was identified as an authorizer, pre-approve that part of the request.



A chronological sequence for authorization must also be considered. In manual systems, authorizers are invited to review a request one after another – authorization simulates the movement of a paper request form. In an automated workflow system, it is possible to invite all authorizers at the same time. This is attractive, as it minimizes the total time between request submission and fulfillment. So long as all the required approvals are provided, there is no security benefit to serializing reviews and approvals – it doesn't matter if A approved a request before B, or B before A. In other words, parallel authorization is a best practice.



To ensure prompt response, it generally makes sense to ask several candidate authorizers for approval and treat a request as approved as soon as some minimum subset of them responds. This creates a race to approve, whereby the fastest approvers move a request to fulfillment at the earliest possible time. In other words, best practice is to ask a group of  $N$  resource owners to approve a request and treat it as approved once  $M$  respond positively, where  $M \leq N$ .



Supporting multiple, concurrent authorizers leads to a possible situation where some authorizers approve a request, while others reject it. The simplest way to handle this possibility is to assume that any rejections that occur prior to the request being approved act as a veto and block the request entirely. Once a request has been approved, by the smallest number of authorizers that is considered acceptable, the request should be closed and all remaining authorizers notified of this fact. This best practice eliminates uncertainty as to the state of a request, while giving authorizers the right to object to one another's approvals, so long as they act in a timely manner. This approach also encourages prompt response by authorizers – if you mean to block a request, then you must review it quickly.



## 8 User and Entitlement Management Processes

A user provisioning system, such as that illustrated in [Figure 3](#) on [Page 7](#), is intended to reduce administration complexity. This is done through implementation of one or more of the following business processes described in [Section 5](#) on [Page 8](#).

The following sections describe each process, when and how to use it, and – equally importantly – when it will not work well.

### 8.1 Identity synchronization

Detect changes to identity attributes, such as phone numbers or department codes, on one system and automatically make matching changes on other systems for the same user.

#### 8.1.1 When to use

Where multiple systems contain the same identity attributes and where that information is updated in a reliable and timely manner on at least one of them.

#### 8.1.2 Scope

Impacts every user that has an account on at least two systems, and where at least one of those systems gets reliable and timely updates to identity attributes.

#### 8.1.3 How to use

Periodically read identity attributes from all systems, find discrepancies, accept data from “more trustworthy” systems and push it out to systems that are “less trustworthy” for the same information.

#### 8.1.4 Pitfalls to avoid

1. Clearly define how “trustworthy” every system is for each identity attribute (phone number, e-mail address, etc.).
2. Recognize that different systems may be more reliable in regards to different information. For example, a white pages application may be a reliable source for phone numbers, the e-mail system may be a reliable source for e-mail addresses and the HR system may be a reliable source for department numbers.



3. Do not make the identity management system itself the “most trustworthy.” At first, this might make sense, since this allows for self-service updates to any information. Unfortunately, this will also gradually make the self-service mechanism authoritative for everything, which is not desirable. Instead, implement a mechanism that allows the identity management system to temporarily override identity attributes that came from an otherwise trustworthy system.

## 8.2 Auto-provisioning and automatic deactivation

Detect new users on an authoritative system (such as HR) and automatically provision those users with appropriate access on other systems and applications.

Detect deleted or deactivated users on an authoritative system and automatically deactivate those users on all other systems and applications.

### 8.2.1 When to use

Auto provisioning and automatic deactivation are effective if and only if:

- A system of record exists.
- That system of record has accurate information about users.
- Updates to the system of record are timely.

If any one of these conditions cannot be met, this feature should not be used.

Where the system of record only relates to certain classes of users, automation will only be effective for those types of users. For example, where the only reliable and timely system of record is HR, auto-provisioning will work for employees but usually not for contractors, vendors, etc.

Finally, auto-provisioning can only be used at the level of granularity of the data available in the system(s) of record. Referring to the previous example, if the HR application only tracks user names, hire date and termination date, then it will not be possible to assign roles to users based on HR data.

### 8.2.2 Scope

In most organizations, the system of record is the human resources (HR) application. Also in most organizations, data in this system is applicable only to employees and is coarse-grained – there is no information about contractors and employee data is very basic. Consequently, in such organizations, auto-provisioning should only be used to:

- Automatically provision basic systems access to new users. For example, network login IDs, e-mail accounts and Internet access can be setup for all new employees.
- Automatically deactivate all access for terminated employees.

Setup and deactivation of contractors are usually handled separately, because contractors are not represented in the HR system.

Fine-grained entitlements are usually assigned using separate processes, because the HR feed cannot predict user needs with any accuracy.

### 8.2.3 How to use

Automation begins with a review of the data quality, timeliness, scope and granularity in each system of record.


Once the type and quality of reference data have been reviewed and accepted, the next step is to identify what data changes in the system of record are relevant and to map those changes to target systems.


Next, a data feed is configured to monitor each system of record and detect changes.

Finally, transformations are defined, mapping data from the format in which it appears (HR – e.g., employee numbers) to the format needed on target systems (e.g., login IDs, e-mail addresses, etc.).

### 8.2.4 Pitfalls to avoid

HR systems sometimes include a job code or similar field to represent the user's role in an organization. This is suggestive of role-based access control (RBAC), where roles are mapped to sets of entitlements and users are provisioned with some or all of the entitlements they will need based on their role.

RBAC is an effective mechanism for large populations of users that perform the same job and consequently need consistent security entitlements. These are typically “front-line” users – retail point of sale, bank tellers and loans officers, etc. 


RBAC may not be cost effective where users have unique requirements. High-value, high-risk employees and contractors are often unique and are consequently not well served by RBAC. For example, the security needs of a company's chief financial officer (CFO) do not benefit from a role being designed for that employee, since only one user will get the role. 


## 8.3 Self-service requests and delegated administration

Enable users to update their identity attributes and to request new entitlements (e.g., access to an application or share).

Enable managers, application owners and other stake-holders to modify users and entitlements within their scope of authority.

### 8.3.1 When to use

Self-service requests and delegated administration are effective for knowledge workers, who are comfortable using a computer, and in particular a web browser, to review current information and request changes. 

Self-service is not appropriate for populations of users who do not have easy access to a computer, who are not comfortable using one, or who require significant training prior to use of each new application. 

### 8.3.2 Scope


Users, their peers and their managers are the most reliable sources of information about changing business needs. It makes sense to enable users to request specific entitlements, such as new roles, accounts and group memberships. It also makes sense to delegate the maintenance of identity attributes – full name, phone number, etc. to users themselves.

While automation is frequently effective for coarse-grained management of employees, delegated administration is often the only option for other classes of users – contractors, vendors, etc. – for whom a system of record may not be available.

Delegated administration makes sense in organizations where managers or IT administrators working at different locations or business units have both the responsibility and expertise to manage users in their own areas.

### 8.3.3 How to use

Do:

1. Make the request system as simple and user friendly as possible.
  2. Advertise the availability of the system to users.
  3. Link to the system from application login prompts and other screens where users might attempt to access applications or data and be prevented because they do not have the appropriate entitlements.
  4. Limit the set of entitlements that a user can request to those that are reasonable for that requester and/or recipient. It makes no sense to allow someone to ask for something which will definitely be rejected later.
- 

5. Evaluate security policies for all requests and pre-emptively reject requests that would trigger a violation (e.g., SoD or similar).

To deploy a self-service request system and/or a delegated administration system, one must address the following design variables:



- Which users will be allowed to make requests.
- Can any user make requests on behalf of any other user? If not, what are the limits relating requesters to recipients?
- Can any user ask for any entitlement? If not, what are the limits relating requesters to entitlements?
- What kinds of requests (create/hire? delete/terminate? change?) will any given requester be allowed to make?

#### 8.3.4 Pitfalls to avoid

Organizations are often tempted to organize the set of requestable resources into a hierarchy. This is often undesirable because users don't know where in the hierarchy to find the resource they are interested in. It is better to offer a search mechanism rather than a "tree view."



## 8.4 Authorization workflow

Validate all proposed changes, regardless of their origin and invite business stake-holders to approve them before they are applied to integrated systems and applications. Please refer to [item 7.1.4](#) on Page 14 for more on this.

### 8.4.1 When to use

Whenever a user provisioning system may accept requests that are not automatically approved, authorization is required. This includes self-service requests for new entitlements and delegated administration requests, by managers or application/data owners.

### 8.4.2 Scope

Every change processed by a user provisioning system which does not either come from a 100% trustworthy source or which represents no business risk, should be authorized before it is fulfilled.

Authorization normally impacts managers, application or data owners and security officers, any or all of whom may have to review and approve or reject change requests.

### 8.4.3 How to use


To deploy an enterprise-scale authorization process, one must address the following design variables:


- How will each type of request be validated (i.e., syntax validation of input fields, code lookups and consistency checks).
- Who are appropriate authorizers for each type of request? Typically some combination of managers in the recipient's chain of command plus application owners is used.
- What is the expected response time from authorizers? When should reminders be sent? When should requests be escalated from unresponsive authorizers to alternate, replacement authorizers?
- What parts of a request are authorizers allowed to see? For example, social security numbers and the like may be part of a request, but not appropriate to display.
- What parts of a request are authorizers allowed to modify? If one authorizer approves a request and a second authorizer modifies it, this might invalidate the earlier approval.
- Define authorizers in terms of groups and identify how many of each group must approve a request before it is considered to be ready for fulfillment.

Each of these design considerations must be resolved before deploying the authorization system – and in many cases before deploying the user provisioning system in general.

#### 8.4.4 Pitfalls to avoid

In a realistic, full-scale user provisioning deployment, there may be hundreds of different kinds of requests – to create, modify or delete accounts on a variety of applications. Because of this scale, it is too expensive to define a separate authorization process for every kind of request. Who will draw hundreds of flowcharts? Who will maintain them?

Instead, it makes sense to define globally-applicable logic for routing requests to the appropriate authorizers, based on the contents of each request – requester, recipient, resources, type of change, etc. 

Authorizers are just human beings, so are unreliable actors in a process which must, in its totality, be reliable. It is essential to allow for the possibility that authorizers may not respond to invitations to approve a request – on time or at all. In practical terms, this means: 

1. Invite more authorizers than are actually needed (invite M, require approval by N, where  $M \geq N$ ).
2. Allow authorizers to temporarily delegate their responsibility, for example when they expect to be unavailable for some time.
3. Send automatic reminders to non-responsive authorizers.
4. After sending a few reminders, escalate from non-responsive authorizers to alternates.
5. If an authorizer is known to be out of the office (e.g., “out of office” message is set on the e-mail system) then escalate before the first invitation is sent.

## 8.5 Consolidated reporting

Provide data about what users have what entitlements, what accounts are dormant or orphaned, about change history, etc. across multiple systems and applications.

### 8.5.1 When to use

Every organization that deploys a user provisioning system should take advantage of its ability to report on identity and entitlement data across systems.

### 8.5.2 Scope

Every user, identity attribute and entitlement managed by the system should be visible in reports.

### 8.5.3 How to use

There are two broad scenarios where IAM reports are helpful:

- On a one-off basis, when trying to answer a specific question. This may be due to an audit or an investigation of an access problem.
- On a recurring or scheduled basis:
  - To identify users who violate policies, such as SoD rules.
  - To identify users with access to sensitive data or applications.
  - To monitor the performance and utilization of the IAM system (e.g., number of requests processed, accounts created/deleted, etc.).

### 8.5.4 Pitfalls to avoid

Data about users, identity attributes and entitlements should be stored in a normalized, relational database with a well documented schema. This makes it possible to develop custom reports that augment those built into the IAM system.



## 9 Internal Controls

Both corporate governance and privacy protection depend on strong security over applications and IT infrastructure. Without such security, internal controls cannot be relied upon and regulatory compliance cannot be assured.

IT security depends heavily on an infrastructure of user authentication, access authorization and audit, commonly referred to as AAA. AAA, in turn, depends on accurate and appropriate information about users – who are they, how are they authenticated and what can they access?

It is in managing these entitlements where organizations have problems. There are too many users, accessing too many systems and they keep moving as a result of hiring, transfer and termination business processes.

AAA infrastructure is nothing new and has been built into every multi-user application for decades. The problem is that a growing number of systems and applications, combined with high staff mobility, have made it much harder to the manage passwords and entitlements on which AAA rests.

With weak passwords, unreliable caller identification at the help desk, orphan accounts, inappropriate security entitlements and mismatched login IDs, AAA systems often wind up enforcing the wrong rules. The weakness is not in the authentication or authorization technology – it's in the business process for managing security entitlements and credentials.

To address problems with AAA data, it is essential to implement robust processes to manage security, so that only the right users get access to the right data, at the right time.

This is accomplished with:

- Better control over how users acquire new entitlements and when entitlements are revoked.
- Correlating user IDs between systems and applications, so that audit logs can be related to real people.
- Periodic audits of entitlements, to verify that they remain business-appropriate.
- Logging of both current and historical entitlements, to support forensic audits.
- Stronger passwords and more robust authentication in general.

## 9.1 Using Roles to Grant Appropriate Entitlements

Role-based access control (RBAC) is an approach to managing entitlements, intended to reduce the cost of security administration, ensure that users have only appropriate entitlements and to terminate no-longer-needed entitlements reliably and promptly.

In the context of a single system or application, RBAC means granting privileges directly to roles and attaching users to roles. Users acquire privileges through role membership, rather than directly. Within a single system, roles are sometimes called security groups or user groups.

Single-system RBAC is a time tested and successful strategy, as it allows administrators to group users, group privileges and attach groups of privileges to groups of users, rather than attaching individual privileges to individual users.

User provisioning systems extend RBAC beyond single applications. Roles in a user provisioning system are sets of entitlements that may span multiple systems and applications. The key element of roles is to replace many technical entitlements with fewer roles that business users can understand. Business users can then a reasonable determination of which users should have which roles. This implicitly specifies which users should have which technical entitlements.

Roles consist of entitlements – login accounts and security group memberships. Roles are often also nested – i.e., one role can contain others. Nesting roles can reduce the cost of role administration.

Using roles, it is possible to:

1. Grant appropriate entitlements to new users, by assigning the correct role.
2. Remove old entitlements when a user's role changes.
3. Make it easier for business users to request the entitlements they need, by aggregating them into easy-to-understand roles.

A best practice is to leverage RBAC for:

1. Coarse-grained, basic entitlements for all users.
2. Fine-grained, full entitlements for users who belong to large communities with identical needs.

RBAC can technically be used to manage the entitlements assigned to every user, but it is not normally cost effective to define a new role for every user with unique requirements.



## 9.2 Enforcing Segregation of Duties Policies

Segregation of duties (SoD) policies allow organizations to define toxic combinations of entitlements, which no one user should possess. The most common business driver for these policies is fraud prevention – i.e., ensuring that fraud cannot be committed without collusion by at least two users.

An effective SoD engine has several components:

- **Policy definitions:**

- Define sets of two or more entitlements that should not be held by a single user.
- Should support different types of entitlements, such as roles, login accounts and groups.
- Should support large sets, such as “no user should have more than 2 of these 30 entitlements.”

- **Approved exceptions:**

There are inevitably situations where an SoD policy should, legitimately, be violated. It should be possible to define approved exceptions to SoD rules.

- **Proactive enforcement:**

Change requests that pass through an IAM system should be subject to SoD policy checking. Changes that would trigger an SoD violation should be blocked at source.

- **After-the-fact detection:**

Many users and entitlements will exist before the IAM system is deployed or before a given SoD policy is defined. Moreover, system administrators may assign entitlements to users outside the IAM system. These scenarios mean that not all SoD violations can be prevented – some have to be detected after the fact and remediated manually.

An effective SoD engine should detect violations even if the policy is stated in terms of roles but the violation is in terms of lower-level entitlements – or vice-versa.

Consider the following variations, where R1 is a role that consists of entitlements Ea and Eb and R2 is a role that consists of entitlements Ec and Ed:

SoD policy	User has	User requests	Why this is a violation
R1 and R2 are mutually exclusive.	R1	R2	Direct violation.
(as above)	R1	Ec, Ed	User would <i>effectively</i> get R2.
Eb and Ec are mutually exclusive.	R1	R2	User has Eb from R1, would get Ec from R2.
(as above)	Ea, Eb	R2	User has Eb directly, would get Ec from R2.
(as above)	Ea, Eb	Ec	User has Eb directly, would get Ec.



Some best practices for SoD enforcement are:

1. Engage financial and compliance officers to help define appropriate SoD policies.
2. Enforce SoD rules proactively, to prevent any new violations.
3. Periodically scan existing entitlements for SoD violations that occurred before an SoD rule was defined or outside the scope of the user provisioning system.
4. Verify that the SoD engine can detect effective violations, not just explicit ones (see above for details).

### 9.3 Periodically Reviewing and Correcting Entitlements

Regulatory compliance requirements and security policies increasingly demand that organizations maintain effective controls over who has access to sensitive corporate information and personal data about employees and customers:

- Systems must limit access to just the right users, at just the right time.
- Organizations must be able to provide auditable evidence that these controls are in place and effective. Section 404 of Sarbanes-Oxley specifically states that management must assess the effectiveness of internal controls on an annual basis.
- Organizations must be able to report which internal users currently have and had in the past, access to sensitive data.

Meeting these requirements can be challenging as users often have unique and changing business responsibilities, thus making their entitlements difficult to model using formal roles and rules.

The difficulty in modeling complex, heterogeneous entitlements is compounded by the fact that although users accumulate entitlements over time, they rarely ask IT to terminate old, unneeded rights. Moreover, it is difficult to predict when, after a change in responsibilities, a user will no longer function as a backup resource for his old job and so old entitlements can be safely deactivated.

These challenges together mean that it is difficult to model all of the entitlements that users need across multiple systems and applications at a single point in time and likely impossible to model those needs for thousands of users, over multiple systems, over an extended period of time.

Access certification is a process where business stake-holders are periodically invited to review entitlements, sign-off on entitlements that appear to be reasonable and flag questionable entitlements for possible removal.

There are several components to access certification:

- **Discovery:**

Before entitlements can be reviewed, they have to be collected from systems and applications and mapped to users. Technical identifiers should be replaced by human-legible descriptions that reviewers will understand. Since entitlements change all the time, discovery should be a regularly scheduled, automated process, not a one-time data load.

- **Who performs the reviews?**

Options include managers – asked to review their subordinates, application or data owners – asked to review lists of users who can access their applications or data or security officers – asked to review high risk entitlements.

- **When are reviews performed?**

The frequency may vary with the business risk posed by the entitlements in question.

- **What kinds of entitlements are reviewed?**

The highest level review is of employment status – should the user in question still have access to any systems? Slightly more granular is a review of roles – should the user in question still have these roles? At the lowest level of granularity are basic entitlements – should the user in question have a login ID on this system or belong to this security group?

- **Which entitlements warrant a review?**

Not every entitlement poses a significant business risk. User membership in the social committee mailing list is not really worth reviewing, for example. Some determination must be made of the risk level posed by each entitlement, as this forms the basis for deciding whether to review it and how often.

- **What happens to rejected entitlements?**

Reviewers may flag entitlements as inappropriate, in which case something should be done. Does this raise a work order in an IT issue management system or trigger a connector to revoke the entitlement immediately? Should further reviews take place before the entitlement is reviewed?

Some best practices for access certification are:



1. Go through the access certification and entitlement cleanup process at least once before starting to define roles. The cleanup will make it easier to identify sets of users with identical security entitlements.
2. Extract entitlement data from integrated systems regularly – e.g., every 24 hours. This will allow reviewers to comment on current, actual entitlements rather than out-of-date data.
3. Give high priority to defining clear, business-friendly descriptions for every entitlement.
4. As roles are defined, enable reviewers to approve the roles assigned to a user, rather than the user's (more numerous) fine-grained entitlements.
5. For small applications, invite application owners to review users with access.
6. For large applications, invite managers to review their subordinates' access, since application owners will not know the users personally.
7. Invite managers to validate the employment status of all users regularly (e.g., quarterly) to eliminate orphan users.

## 10 Integrations with Systems and Applications

Medium to large organizations typically have thousands of users who need access to hundreds of applications.

Even if integrating a user provisioning system with an application is very fast and inexpensive – say 1 day of effort, including integration, testing, ID mapping, etc. – it would still take hundreds of person-days of effort to integrate every application. Waiting for these integrations to be completed before rolling out the user provisioning system would unacceptably increase the cost of the system and the delay before it starts to produce value.

In most organizations, the mix of systems and applications includes a few widely-used systems and hundreds of smaller applications, which have relatively few users:

1. **Enterprise systems:** These typically include directories, mainframes, ERP systems, e-mail systems, etc. There are rarely more than 10 or 20 such systems.
2. **Small applications:** These are typically departmental or specialized and may include financial and budgeting systems, engineering applications, systems used within IT, etc.

To maximize the value of a user provisioning system and to minimize delay between acquisition and production use of the system, it makes sense to:



1. Integrate with the major, enterprise-wide systems and applications during the initial phase of deployment.
2. Support requests for access to smaller applications in the user provisioning system, so that uniform request and approval processes, along with policy enforcement can be applied to every application.
3. Ask (human) system administrators to fulfill approved requests for access to non-integrated applications.
4. Gradually add application integrations, prioritized by request volume.

This approach creates a “one stop shopping” experience for requesters and authorizers and supports uniform audit processes, SoD policy enforcement and access certification, regardless of the integration status of any given application.

Taking things one step further, it makes sense to implement limited integrations with as many applications as possible, as early as possible. In practical terms, this means that the user provisioning system should be configured to list accounts on each application automatically, so that it has up-to-date data about what users already have access to each application. This is essential if SoD and access certification processes are to have any meaning – otherwise, what is being certified?

An open question is where should the workflow processes that invite system administrators to make changes reside? One option is to place these workflow processes on an existing IT infrastructure management platform, such as BMC Remedy or HP Service Manager. Another approach is to track requests for action, acknowledgement of tasks and indication of completion right in the user provisioning system.

Either approach can work, but in any case the process should support:



1. Multiple system administrators per application.
2. Reminders to non-responsive administrators.
3. Escalation from non-responsive administrators to alternates.
4. Feedback from administrators, providing information generated during the user create/delete process, such as system-generated unique IDs.

## 11 Summary

User provisioning systems create value by lowering IT support costs, improving user service and strengthening network security. They do this by:

1. Propagating changes to identity and entitlement data from one system to another – identity synchronization, auto-provisioning and auto-deactivation.
2. Enabling business users to request, review and approve changes to identity attributes and entitlements.
3. Enforcing security policies using roles, SoD rules, standard naming conventions and more.
4. Reporting on users and entitlements, both current state and historical.

By taking advantage of best practices presented in this document, organizations will be able to minimize the cost of deploying a user provisioning system while maximizing its value.

# APPENDICES

## A Identity Manager Overview

### Overview:

Hitachi ID Identity Manager is a complete **identity management** solution that automates and simplifies the tasks of managing users and entitlements across multiple systems and applications throughout the user lifecycle. Organizations depend on **Identity Manager** to ensure that users get appropriate access rights promptly and are deprovisioned reliably and completely.

Identity Manager implements the following business processes to drive changes to users and entitlements on systems and applications:

- **Automation:** grant or revoke access based on data feeds.
- **Synchronization:** keep identity attributes consistent across applications.
- **Self-service:** empower users to update their own profiles.
- **Delegated administration:** allow business stake-holders to request changes directly.
- **Certification:** invite managers and application owners to review and correct entitlements.
- **Workflow:** invite business stake-holders to approve or reject requested changes.

### Features:

Identity Manager enables automated, self-service and policy-driven management of users and entitlements with:

- **Auto-provisioning and auto-deactivation:**

Identity Manager can monitor one or more systems of record (typically HR applications) and detect changes, such as new hires and terminations. It can make matching updates to other systems when it detects changes, such as creating login accounts for new employees and deactivating access for departed staff.

- **Identity synchronization:**

Identity Manager can combine identity information from different sources – HR, corporate directory, e-mail system and more into a master profile that captures all of the key information about every user in an organization. It can then write updates back to integrated systems, to ensure that identity attributes are consistent. This feature is used to automatically propagate updates to data such as names, phone numbers and addresses from one system to another.

- **Self-service updates:**

Users can sign into the Identity Manager web portal and make updates to their own profiles. This includes changes to their contact information and requests for new access to applications, shares, folders, etc.

- **Delegated administration:**

Business stake-holders, such as managers, application owners and data owners can sign into the Identity Manager web portal and request changes to security entitlements. For example, a manager might ask for application access for an employee or schedule deactivation of a contractor's profile.

- **Access certification:**

Business stake-holders may be periodically invited to review the users and security entitlements within their scope of authority. They must then either certify that each user or entitlement remains appropriate or flag it for removal. Access certification is an effective strategy for removing security entitlements that are no longer needed.

- **Authorization workflow:**

All change requests processed by Identity Manager, regardless of whether they originated with the auto-provisioning engine, the identity synchronization engine, with self-service profile updates or with the delegated administration module may be subject to an authorization process before being completed. The built-in workflow engine is designed to get quick and reliable feedback from groups of business users, who may be individually unreliable. It supports:

- Concurrent invitations to multiple users to review a request.
- Approval by N of M authorizers (N is fewer than M).
- Automatic reminders to non-responsive authorizers.
- Escalation from non-responsive authorizers to their alternates.
- Scheduled delegation of approval responsibility from unavailable to alternate approvers.

- **Policy enforcement:**

Identity Manager can be used to enforce a variety of policies regarding the assignment of security entitlements to users, including:

- Role based access control, where security entitlements are grouped into roles, which can be assigned to users.
- Segregation of duties, which defines mutually-exclusive sets of security entitlements.
- Template accounts, which define how new users are to be provisioned.
- Rules for the composition of new IDs, such as login IDs, e-mail addresses, OU directory contexts and more.

- **Reports:**

Identity Manager includes a rich set of built-in reports, designed to answer a variety of questions, such as:

- What users have entitlement X?
- What entitlements does user Y have?
- Who authorized entitlement Z for user W?
- When did user A acquire entitlement B?
- Who requested and who authorized entitlement B for user A?
- What accounts have no known owner (orphaned)?
- What users have no accounts (empty profiles)?
- What accounts have recent login activity (dormant)?
- What users have no active accounts (dormant)?

- **Automated connectors and human implementers:**

Identity Manager can be integrated with existing systems and applications using a rich set of over 110 included connectors. This allows it to automatically provision, update and deprovision access across commonly available systems and applications.

Organizations may opt to integrate custom and vertical-market applications with Identity Manager by using the included flexible connectors. Alternately, the built-in “implementers” workflow can be used to invite human administrators to make approved changes to users and entitlements on those systems.

- **Unified management of logical access and physical assets:**

Identity Manager includes an inventory tracking system, making it suitable for managing requests for physical assets as well as logical access. For example, types and inventories of building access badges, laptops, phones and other devices can be tracked, requested, authorized and delivered using Identity Manager.

### Benefits:

Identity Manager strengthens security by:

- Quickly and reliably removing access to all systems and applications when users leave an organization.
- Finding and helping to clean up orphan and dormant accounts.
- Assigning standardized access rights, using roles and rules, to new and transitioned users.
- Enforcing policy regarding segregation of duties and identifying users who are already in violation.
- Ensuring that changes to user entitlements are always authorized before they are completed.
- Asking business stake-holders to periodically review user entitlements and either certify or remove them, as appropriate.
- Reducing the number and scope of administrator-level accounts needed to manage user access to systems and applications.
- Providing readily accessible audit data regarding current and historical security entitlements, including who requested and approved every change.

Identity Manager reduces the cost of managing users and security entitlements:

- Auto-provisioning and auto-deactivation leverage data feeds from HR systems to eliminate routine, manual user setup and tear-down.
- Self-service eliminates IT involvement in simple updates to user names, phone numbers and addresses.
- Delegated administration moves the responsibility for requesting and approving common changes, such as for new application or folder access, to business users.

- Identity synchronization means that corrections to user information can be made just once, on an authoritative system and are then automatically copied to other applications.
- Built-in reports make it easier to answer audit questions, such as “who had access to this system on this date?” or “who authorized this user to have this entitlement?”