

## Beyond Roles:

A Practical Approach  
to Enterprise User Provisioning



Enterprise-scale organizations employ large numbers of internal users, with different access requirements spanning large numbers of systems, directories and applications. The dynamic nature of modern enterprises demand that organizations efficiently and securely provision and deactivate systems access to reflect rapidly changing user responsibilities.

This document introduces a strategy for large-scale enterprise user administration. This strategy complements the traditional role-based approach with user-issued security requests combined with periodic audits. Using this approach, new privileges are granted to users in response to user-entered requests, rather than being predicted by an automatic privilege model. Excessive user privileges are periodically identified and cleaned up using a distributed, interactive user rights review and certification process.

# Contents

- 1 Introduction** **1**
  
- 2 Background** **2**
  - 2.1 The Business Problem . . . . . 2
  - 2.2 Access Control Infrastructure . . . . . 2
  - 2.3 Consolidated User Provisioning . . . . . 3
  
- 3 Privilege Modeling** **4**
  - 3.1 Role-Based Access Control . . . . . 4
  - 3.2 Roles and Rules . . . . . 4
  - 3.3 Benefits of a Privilege Model Strategy for User Administration . . . . . 5
  - 3.4 Drawbacks of a Privilege Model Strategy to User Administration . . . . . 5
  - 3.5 Feasibility of the Privilege Model Approach . . . . . 6
  
- 4 From Theory to Reality** **7**
  - 4.1 Deployment Problems . . . . . 7
    - 4.1.1 User Diversity . . . . . 7
    - 4.1.2 Arithmetic Explosion . . . . . 7
    - 4.1.3 Changing Responsibilities . . . . . 8
    - 4.1.4 Business Reorganization . . . . . 9
    - 4.1.5 Configuration and Maintenance Cost vs. Return On Investment . . . . . 9
  - 4.2 Special Cases: Where Privilege Models Work . . . . . 9
  
- 5 User Provisioning Using Requests and Reviews** **11**
  - 5.1 A Request-Based User Administration Strategy . . . . . 11

5.2	Self-Service Workflow Engine . . . . .	11
5.3	Limited Use of Roles . . . . .	12
5.4	Privilege Accumulation . . . . .	12
5.5	User Privilege Reviews . . . . .	13
5.6	User Access Certification . . . . .	13
5.7	Summary . . . . .	14
<b>6</b>	<b>Conclusions</b>	<b>15</b>
<b>7</b>	<b>References</b>	<b>16</b>

# 1 Introduction

Enterprise-scale organizations employ large numbers of internal users, with different access requirements spanning large numbers of systems, directories and applications. The dynamic nature of modern enterprises demand that organizations efficiently and securely provision and deactivate systems access to reflect rapidly changing user responsibilities.

User provisioning systems are intended to streamline and secure enterprise-wide user administration by consolidating these processes into a shared infrastructure.

Previous approaches to consolidated user provisioning have focused on constructing and maintaining a formal model of user privileges, including roles and rules, that predicts what accounts and rights should be assigned to any given user, based on user classification and other identity attributes.

In real-world deployments, formal models have not scaled well, because many users are unique and consequently there is no leverage to be gained by grouping them into roles or generalizing their access rights with rules. Indeed, role engineering projects are often abandoned or severely curtailed after significant effort and expense.

This document introduces a strategy for large-scale enterprise user administration. A traditional pre-defined role-based approach can practically be applied only to standard, static roles. The strategy offered in this document offers a complementing approach to automated privileges management for unique and/or dynamic roles. It is based on user-issued access requests combined with periodic audits.

Using this approach, new privileges are granted to users in response to user-entered requests, rather than being predicted by an automatic privilege model. Excessive user privileges are periodically identified and cleaned up using a distributed, interactive user rights review and certification process.

The remainder of this document is organized as follows:

- **Privilege Modeling**

An introduction to privilege modeling as a strategy for automated user administration.

- **From Theory to Reality**

Where privilege modeling does work in practical deployments, where it doesn't and why.

- **User Provisioning Using Requests and Reviews**

A strategy to user administration is introduced, based on user-submitted requests for new and changed privileges. A periodic access certification process is also introduced, to address the problem of privilege accumulation.

- **References**

Supporting materials.

## 2 Background

### 2.1 The Business Problem

Modern organizations and in particular large enterprises with tens of thousands of users, are deploying an increasingly wide array of IT infrastructure. This infrastructure is accessed not only by employees, but also by contractors, partners, vendors, customers and in some cases even competitors.

As the complexity of IT infrastructure increases and as the the number and diversity of people who must access it grows, it has become apparent that traditional tools and processes to manage security are breaking down. When security administration breaks down, companies fail to:

- Provision users with systems access in a timely manner.
- Create users with access privileges appropriate to their jobs.
- Terminate access when it is no longer required, in a timely and reliable manner.

These problems lead to serious business costs: reduced efficiency, delayed business results and vulnerability to internal and external security attacks.

Security exposures due to improperly managed user privileges have become an especially serious problem, since they put organizations out of compliance with regulations such as Sarbanes-Oxley, Gramm-Leach-Bliley, HIPAA and FDA 21-CFR-11, all of which call for effective internal controls.

### 2.2 Access Control Infrastructure

Most information systems manage access to data and to processes by identifying individual users, by controlling what each user can see and do and by recording user actions.

The access control infrastructure in information systems is broken down into four distinct parts:

- **User identification**, typically using login IDs, employee numbers, etc.
- **User authentication**, typically implemented with passwords, security tokens, biometric identification, challenge/response data or a combination of these authentication factors.
- **User authorization**, using access-control lists, group membership, application roles, etc.
- **Audit**, recording user access to systems and data.

Systems that implement security controls normally come with their own tools to manage the three kinds of user data: IDs, authentication factors and authorization rules. For example, Windows NT includes the “User Manager” GUI, Unix systems administrators edit `passwd` files, Oracle DBAs issue SQL commands and SAP administrators use the SAP GUI.

These administration programs are “point solutions,” intended to manage a single system. Problems with user administration result not from these individual tools, but from the proliferation of administrative interfaces, the number of administrators required to operate them and the number of application owners who must authorize use of each system.

There are simply too many people involved in what should be a simple task: grant access today and delete it later.

## 2.3 Consolidated User Provisioning

Third party user provisioning systems – i.e., those that do not come bundled with the various existing applications, operating systems and directories – are intended to address the problem of complexity by consolidating user administration across systems and streamlining user administration business processes.

All user provisioning systems automate some subset of the following business 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.

## 3 Privilege Modeling

*Companies that try and establish roles for an entire enterprise, as opposed to one application or department, could end up with as many roles as there are employees. It's hard to maintain because the business is always changing. So you must start small and look at all of this identity management as evolutionary.*

— Roberta Witty  
Research VP, Security & Privacy  
Gartner

A traditional strategy to enterprise user provisioning is to model user privileges using roles and rules and to manage users by classifying them into roles and by applying attribute-based rules to calculate appropriate user rights.

In the context of a single system, this approach is well established as the best practice. For example, it is normal, practical and effective to manage user privileges in an Oracle database using database roles, in Active Directory using groups or in SAP using Activity Groups.

With role- or group-based administration, where many users belong to each role or group, it is more efficient to assign users to roles/groups and to assign privileges to the role or group, than to assign privileges directly to users. The key is that many users share the same privilege profile. As a result, security privileges are easier to manage across a group of users than they would be if applied directly to individuals.

Privilege modeling is very difficult to implement when applied to large numbers of users who each require access to multiple systems.

In other words, it is both time consuming and costly to create and maintain a role model which defines privileges across many systems and apply it effectively to many unique users.

### 3.1 Role-Based Access Control

The strategy of using a formal model to predict user rights based on role assignment and rules that extend user attributes is sometimes called policy-based provisioning or role-based access control (RBAC).

In a role-based system, privileges are attached to roles and each user is classified into one or more roles. These systems assign privileges to users strictly through their role membership.

### 3.2 Roles and Rules

The challenge with a role-based approach to modeling user privileges is that as departments, systems and applications are added to the scope of the user provisioning system, user diversity is exposed, and the number of roles required to adequately model security rights grows.

Implementers of RBAC systems realize that some privileges and user attributes are unlikely to be exactly

alike for every user in a role. Consequently, rules are introduced as dynamic logic to extend static roles. For example, rules might determine the appropriate home directory path for a user or create a mail folder on a suitable mail server.

Basically, roles define static collections of privileges that apply uniformly to many users. Rules extend this static model by examining user attributes – such as department code or location code – and specifying additional details – such as mail server location – based on these user-specific variables.

With a combination of roles and rules, the full set of privileges assigned to a given user is calculated as follows:

Privileges = User-Role × Role-Definition + Rules-Function( User-Attributes )

In an organization that successfully deploys a roles- and rules-based user provisioning system, it is sufficient to attach a user to one or more roles, and to define a few attributes, to calculate all of the privileges that the user should have.

### 3.3 Benefits of a Privilege Model Strategy for User Administration

The privilege modeling (RBAC) approach to user provisioning promises some appealing benefits:

- Where many users require the same access rights, those rights can be defined once and applied to multiple recipients. This is cost effective.
- In the event that the set of privileges required by a whole set of users changes, the role definition can be adjusted and the changes can then be applied to many users at once.
- By changing a user's role classification, it is possible to quickly and reliably revoke some privileges and grant new ones.

### 3.4 Drawbacks of a Privilege Model Strategy to User Administration

The benefits of a user provisioning system based on roles and rules come at a cost:

- The privilege model, consisting of roles and rules, must be created when the project starts and maintained in perpetuity.
- All users must be classified into at least one role.
- Any deviation between the security rights predicted by the role model and actual user rights must be reviewed (initially and periodically) and possibly approved.

In short, managing the role model itself can be just as costly as managing user privileges directly.

### 3.5 Feasibility of the Privilege Model Approach

This document raises three basic questions about the privilege modeling strategy to user administration:

1. Can privilege modeling really be deployed in the context of an enterprise-scale user provisioning system for non-standard roles?
2. If it can, then under which circumstances?
3. If it cannot, then which alternative strategy can work?

## 4 From Theory to Reality

### 4.1 Deployment Problems

This section describes some of the practical difficulties in implementing a system for managing user privileges based on a formal model and specifically based on user classification into roles (RBAC), in a large organization.

The large number of roles required to model security rights in a large, complex enterprise, combined with the need to frequently update the role model to reflect changing business requirements, can make enterprise-scale role engineering costly and time consuming.

As a result, many large role engineering projects either have their scope reduced or are terminated after significant cost overruns.

#### 4.1.1 User Diversity

In practice, large organizations may have large numbers of unique users. This makes it difficult to construct a role model, since each unique user requires their own role. Unique user requirements may be a product of employees and contractors whose responsibilities have evolved over time or simply whose function is unique in the organization.

Diversification of user access requirements poses a serious problem for a policy-based provisioning system:

- It is difficult to define a “reasonably small” set of roles that account for the access privileges of the bulk of the user population.
- It is impractical to define as many roles as there are users – that is a degenerate case equivalent to manual user administration.

Using rules as well as roles does reduce the number of roles required to model all users. However, introducing rules to supplement roles does not sufficiently reduce the number of roles needed to model user rights.

If this problem cannot be addressed, a system for user privilege management based primarily on role-based user provisioning is likely to fail during deployment. The implementation team will simply get bogged down trying to define all the relevant roles and trying to classify users into those roles.

#### 4.1.2 Arithmetic Explosion

User diversity can be thought of as arising when many systems are brought into the scope of the user provisioning system.

Consider a simple example, where users objects are managed on a single system: the network OS (Windows, NetWare, etc.). In this example, most users will start out with just one role: basic network and e-mail

access. Additional roles will be developed to account for departmental and regional security groups, but the number of roles will most likely be manageable.

When a second system is added to the same deployment, such as a corporate mainframe, the various types of users on that system will have to be added to the role model. Users that had a common role on the network OS will have different sets of privileges on the mainframe. Conversely, users with the same set of privileges on the mainframe may have different access profiles on the network OS. If there were  $N$  types of users on the network OS and  $M$  types of users on the mainframe, then approximately  $N \times M$  roles will be needed to describe all classes of users in a manner that spans both systems.

What happened in the above example is that a single target system was added and some fraction of the roles that were previously defined had to be broken up into new, more fine-grained, roles to allow for the variation that this exposed.

In practice, every time that a target system is added to a role-based user provisioning system, some of the existing roles have to be “exploded” into multiple new, more fine-grained roles. This creates an arithmetic explosion in the number of role definitions, which can lead to situations where there are more roles than users.

This can be modeled mathematically. If adding a target system causes just  $\frac{1}{10}$  of the existing roles to be replaced by 10 roles each and the first system has just 20 roles, the progressive explosion in roles will be as shown below:

Number of systems	Calculation	Number of roles
1	Initial number of roles	20
2	$20 - 2 + 20$	38
3	$38 - 4 + 40$	74
4	$74 - 7 + 70$	137
5	$137 - 14 + 140$	263
6	$263 - 26 + 260$	497
7	$497 - 50 + 500$	947
8	$947 - 95 + 950$	1802
9	$1802 - 180 + 1800$	3422
10	$3422 - 342 + 3420$	6500

Obviously defining 6,500 roles is impractical, as is classifying thousands of users into so many roles.

In practical deployments, hundreds of systems may be managed by a user provisioning system, aggravating the problem.

#### 4.1.3 Changing Responsibilities

If a project to deploy a user provisioning system based on a privilege modeling strategy can overcome the initial problem of defining an adequate set of roles and classifying all users into these roles, it must still contend with the fact that both role definitions and user classification must change constantly.

Routine business events, such as changing the responsibilities of a single employee or extending the function of a department, require that the role model be adjusted. The ongoing nature of such events means that a significant team of expert staff is required to continually maintain the role model.

One approach to the need for constant maintenance of the role model is to delegate role definitions to business users. In practice users have neither the skills nor the inclination to spend time maintaining a technical model of user rights by job code. As a result, dedicated technical staff must be retained to build and maintain the role model at a high cost.

#### 4.1.4 Business Reorganization

Less frequent but more far-reaching business events, such as acquisitions, divestiture, mergers and reorganizations are disruptive to a role-based system. Each of these events can cause massive changes to the privilege model and calls for a significantly larger team to perform remedial role engineering on a one-time basis, under time pressure.

#### 4.1.5 Configuration and Maintenance Cost vs. Return On Investment

The effort required to manage the privilege model (not actual user privileges, just the model of user privileges) can easily be larger than the cost savings achieved by eliminating some or all manual user administration. Moreover, administration of the role model tends to require more skilled (and therefore more expensive) staff than direct administration of user privileges.

## 4.2 Special Cases: Where Privilege Models Work

The preceding sections lay out reasons why a user administration system based on a formal privilege model is difficult and expensive to deploy and why even a successful deployment may nonetheless cost more than manual user administration.

There are, however, some conditions where a role-based system will work well:

1. Within a single system, the number of possible combinations of user rights tends to be small. This is because the arithmetic explosion described in [subsubsection 4.1.2](#) on [Page 7](#) does not happen on a single system. As a result, the number of roles is more manageable, and an RBAC system can be deployed.

In fact, RBAC has long been the best practice inside a single application, database server or operating system.

2. In some organizations, there is a sizable subset of the user population whose privilege requirements are the same and who are therefore suitable to administration using a formal privilege model.

Some examples of user populations suitable to RBAC are:

- In a bank: tellers.
- In a retail outlet: sales clerks.

## Beyond Roles: A Practical Approach to Enterprise User Provisioning

- In an airline: flight attendants.
- In a fast food chain: sales and in-store food preparation staff.

In each of these examples, just a handful of roles are sufficient for modeling the privilege requirements of thousands of users.

Most examples of large and regular user populations seem to be in point-of-sale positions. However, it is noteworthy that in every organization where there is some large and regular user population, there is inevitably another large community of users that are not regular and whose privileges are difficult or impossible to model. This includes the management structure of most organizations, and departments such as accounting, purchasing, engineering, marketing, legal, IT, HR, etc.

The conclusion here is that RBAC is a useful solution for some specific problems – managing users in a single directory or managing uniform user populations – but is not appropriate for the more general problem of managing the access rights of many diverse and dynamic users across many systems.

## 5 User Provisioning Using Requests and Reviews

### 5.1 A Request-Based User Administration Strategy

Because of the challenges in implementing a 100% role-based privilege management system, most organizations also use a request-based system.

Request based systems can be automated, typically with web-based change request forms and e-mail based invitations asking appropriate users to review and authorize changes.

Automated requests are also possible: a data feed from a system of record, such as an HR application, may automatically trigger change requests.

### 5.2 Self-Service Workflow Engine

The most valuable and also most complex component of a request-based user provisioning system is workflow, which must incorporate the following components:

- Identification and authentication of users who can submit change requests.
- Control over what existing user profile data a given requester can see.
- Control over what kinds of changes each requester may submit.
- Form input and validation for user profile change requests.
- Resource selection for profile change requests (enabling requesters to choose systems, account types, security groups and so on).
- Routing of change requests to suitable authorizers for approval.
- Reminders to non-responsive authorizers.
- Escalation from non-responsive to alternate authorizers.
- Delegation of responsibility from one authorizer to another.
- Status review by requesters and authorizers during the request process.
- Integration with a fulfillment service, to automatically approved requests.
- Support for fulfillment of requests by human implementers, where automatic fulfillment is either not possible or not economically feasible.
- Reporting on requests, approvals, queue times, escalations, delegation, etc.

### 5.3 Limited Use of Roles

RBAC (role-based access control) should be used where appropriate and economical – i.e., where there are large numbers of users with identical or very similar entitlement requirements. Other techniques should supplement RBAC where RBAC is too costly to define or maintain:

- Always use roles inside individual applications.
- Use meta-roles (i.e., roles that span multiple applications) to manage the security rights of large sets of users with identical requirements that don't change often.
- Use roles to assign new users with their initial set of entitlements.
- Try to define relatively few, relatively coarse-grained roles rather attempting to capture every entitlement needed by every user with large numbers of fine-grained roles.
- Try to avoid over-reliance on roles when managing access rights of users whose needs are either unique or rapidly evolving.

### 5.4 Privilege Accumulation

A request-based strategy for managing user access rights can create a problem of privilege accumulation.

While users can be counted on to request whatever privileges they need to do their jobs, they are far less likely to submit change requests to deactivate unneeded privileges. As a result, as user responsibilities change over time, users tend to accumulate privileges, rather than adding some and relinquishing others.

Users with more privileges than they need are a clear security problem. In an organization where compliance with privacy protection or corporate governance regulations (Sarbanes-Oxley, Gramm-Leach-Bliley, HIPAA, 21 CFR Part 11, PIPEDA, etc.) is mandatory, privilege accumulation represents an unacceptable risk.

A formal model of user privileges would address the problem of privilege accumulation, but such a strategy can be hard to implement where users are dynamic and/or diverse:

- The number of roles required to model user privileges may be close to the number of users.
- Role definition can be costly and time consuming.
- Assigning appropriate roles to every user can be costly and time consuming.
- Ongoing management of both role definitions and user classification can be more expensive than direct user administration.

Because of these challenges, role-based approaches tend to work well only for static, uniform populations of users, but do not lend themselves to users that change responsibilities very quickly or that have unique access requirements. Unfortunately, the highest risk users in most organizations are exactly those whose privileges are hard to model – back-office users with access to sensitive data.

Approaches beyond formal modeling are required to address the problem of privilege accumulation for these high risk users.

## 5.5 User Privilege Reviews

The simplest method to address the problem of privilege accumulation and the method used by many organizations prior to automating user administration processes, is to periodically audit user privileges and remove those that are no longer relevant.

Audits come with their own challenges, however. One problem is that a single report detailing the privileges of thousands of users across hundreds of systems is not helpful, since there is no one person who could read such a report and make informed decisions about the appropriateness of each user's rights.

Instead of a single, global user privilege audit, it makes more sense to have many local audits. Stakeholders who might be able to reliably comment on the appropriateness of a user's privileges and so could contribute to such local audits, include:

- The user being audited
- The user's manager(s)
- Owners of applications to which the user has access
- Owners of data to which the user has access

Users will generally be disinterested in cleaning up their own privileges and in many cases may actively wish to accumulate privileges. As a result, despite the fact that users should know which of their existing privileges are appropriate, they cannot be relied on to audit themselves. This leaves their managers and application or data owners as suitable stake-holders for a user rights audit.

Because users cannot be trusted to audit themselves, some combination of managers, application owners and data owners must be called on to periodically audit user privileges, to identify inappropriate rights and to initiate a review and approval process prior to deactivating excess privileges.

Since some applications and data sets have large numbers of users, their owners cannot realistically review every user. As a result, in systems that have many users, managers must audit their subordinates – nothing else will scale.

In smaller applications, where there are fewer users, it is reasonable to ask application or data owners to audit the application's users and each user's rights. In these cases, it is reasonable to expect the application owners to personally know who the users are and what kind of access is appropriate for each one.

## 5.6 User Access Certification

In a large organization, there will be many managers, application owners and data owners who must perform periodic audits of user access privileges. It follows that some mechanism is required to ensure that these audits are in fact carried out and performed diligently.

Audits by application and data owners are straightforward – this can be made a core part of the responsibility of these stakeholders and since there are relatively few such stakeholders, ensuring that they complete periodic user privilege audits.

Audits of users by their direct supervisors can be more difficult, since there may be thousands of such supervisors and it is hard to make them all comply with any single directive.

One approach to motivating managers to review the access rights of their direct subordinates is to require a signature at the end of every such review, but to block such signatures until subordinate managers have completed their own reviews. This signature underlies a legal statement by each manager, certifying that the remaining list of that manager's direct subordinates and their privileges, are appropriate.

With this process, an executive such as the CEO or CFO, who wishes to implement strong controls to support a regulatory compliance program, will pressure his direct subordinates to complete their own reviews. They will be unable to sign off until their own subordinates have finished and so a downward pressure through the organization to complete the audit is created. Whereas pressure to perform the user privilege reviews flows downwards from the top of the organization, results of the audit, including cleaned up user rights, flow back up from the lowest-level managers right to the CEO or CFO.

### 5.7 Summary

A request-based system for user administration is much easier to implement than one based on a formal privilege model.

A problem that arises with a request-based system is privilege accumulation. This problem can be addressed using periodic access reviews, cleanup and certification.

## 6 Conclusions

Managing users in a large, dynamic and heterogeneous enterprise IT environment is challenging. Specific business problems that arise from overburdened user administration include:

- Slow activation for new users.
- Privilege accumulation as users move through an organization.
- Slow and unreliable access termination.

In the past, many “industry insiders” have supported automating user administration using a privilege model to address these problems. While this model is attractive in theory, it is very costly and risky to deploy in practice, so does not work well in most cases.

A new strategy for efficient and reliable user administration is presented, based on user-initiated change requests and periodic audits of current user privileges. This approach yields essentially the same end results as a privilege model, but is significantly easier to deploy and maintain.

## 7 References

- **Role Based Access Control:**  
National Institute of Standards and Technology,  
<http://csrc.nist.gov/rbac/>
- **Security Provisioning: Managing Access in Extended Enterprises:**  
Yuri Pikover and Jeff Drake, Information Systems Audit and Control Foundation (ISACA), 2002, ISBN 1-893209-33-4.
- **Role-based access control on a roll:**  
Ellen Messmer, Network World, 07/30/01,  
<http://www.networkworld.com/archive/2001/>
- **Identity Management in the Real World:**  
Deborah Radcliff, CSO Magazine, November 2004,  
<http://www.csoonline.com/read/110104/idmgmt.html>
- **Access Management: Is there a role for RBAC?:**  
Gerry Gebel, Analyst, The Burton Group, July 10, 2003
- **Hitachi ID Identity Manager – User provisioning, RBAC, SoD and access certification:**  
<http://Hitachi-ID.com/Identity-Manager>
- **Hitachi ID Access Certifier – Periodic review and cleanup of security entitlements:**  
<http://Hitachi-ID.com/Access-Certifier>