NSF Middleware Initiative | Brendan Bellina, Editor

internet2-mace-dir-metadirectories-practices-200210.htm | University of Notre Dame

Copyright © 2002 by UCAID and/or the respective authors | October 2002

Comments to: nmi-support@nsf-middleware.org

Development of this document was supported with funding from the University of Notre Dame, Internet2, and the NSF Middleware Initiative (Cooperative Agreement No. ANI-0123937).

Metadirectory Practices for Enterprise Directories in Higher Education

Abstract

This document outlines a set of metadirectory (or *meta-directory*) issues that should be considered in the deployment of enterprise directories and offers accompanying best practices for higher education. The model outlined in this document was assembled from the authors' experiences, discussions among MACE-dir participants, and through interviews with those at other institutions.

This document is a product of the Internet2 Middleware Initiative and the Middleware Architecture Committee for Education (MACE), published under the auspices and with the support of the National Science Foundation Middleware Initiative (NMI). Internet2 is a member of the Enterprise and Desktop and Integration Technologies (EDIT) Consortium, participating in the NMI.

This is one of a growing set of documents created by MACE detailing various issues surrounding the planning, deployment, configuration, maintenance, and security of enterprise directories.

For additional information and related topics and resources see the following sites:

Internet2 Middleware Initiative:	http://middleware.internet2.edu/
MACE:	http://middleware.internet2.edu/MACE/
EDIT:	http://www.nmi-edit.org/
NMI:	http://www.nsf-middleware.org/

Table of Contents

1 Introduction

This document concentrates on the architectural issues that confront the enterprise directory architect, designer, and implementer. The authors resisted the temptation to go down the many side paths. Instead the focus of this document is to provide ways to think about metadirectory processes to help keep the big picture in view while navigating common challenges. Readers who have implemented at least one enterprise directory will certainly recognize the issues. The hope of the authors is that readers will find value in comparing their practices with the ones described here.

The Burton Group coined the term "meta-directory" in the July 1998 paper "Enterprise Directory Infrastructure: Meta-directory Concepts and Functions" as a technology or class of functionality required to build an enterprise directory infrastructure. This definition applies directly to the construction of enterprise directories at institutions of higher learning. These enterprise directories often require the collection and transformation of data about resources from systems of record such as human resources information systems and student information systems.

Unlike directories designed to meet the needs of a single application, such as a campus white pages directory or an email directory, enterprise directories are designed to meet both the immediate needs of existing applications as well as the future needs of applications and services yet-to-come. The result is enterprise directories require a greater focus on the processes for migrating data from systems of record into an enterprise directory and providing that data to other services, systems, and application directories.

2 Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119.

3 Terminology & Usage Examples

ACI - Access Control Instruction

Access control is the mechanism by which you define access. When the server receives a request, it uses the authentication information provided by the user in the bind operation, and the access control instructions (ACI's) defined in the server to allow or deny access to directory information. The server can allow or deny permissions such as read, write, search, and compare. The permission level granted to a user may be dependent on the authentication information provided. (Definition taken from iPlanet Directory Server 4 documentation.)

ACL - Access Control List

A list of Access Control Instructions (ACI's) constitutes an Access Control List.

authentication (***authN***)

Authentication is the process of establishing whether or not a real-world subject is who or what its identifier says it is. Identity can be proven by: Something you know, like a password; Something you have, as with smart-cards, challenge-response mechanisms, or public-key certificates; Something you are, as with positive photo identification, fingerprints, and biometrics. (For more on this topic, see Internet-2 Middleware Authentication website <http://middleware.internet2.edu/core/authentication.html>.)

authorization (***authZ***)

The determination that a request can be honored is known as authorization. (For more on this topic, see Internet-2 Middleware Authorization website <http://middleware.internet2.edu/core/authorization.html>.)

CSO Nameserver (aka ***ph*** and ***qi***)

A directory server product created at the University of Illinois and widely adopted at higher education institutions for white pages functions. CSO Nameserver has little support for privacy, personalization, and security and was not adopted by vendors except for Qualcomm Eudora, a fairly popular email client.

data providers

Systems of record providing data to other systems and processes are called data providers.

data consumers

Systems and processes that retrieve or receive data from the enterprise directory are known as data consumers.

directory

A directory is a specialized database that may contain information about an institution's membership, groups, roles, devices, systems, services, locations, and other resources.

enterprise directory

An enterprise directory is a core middleware architecture that may provide common authentication, authorization, and attribute services to electronic services offered by an institution. See the "Middleware Business Case" http://middleware.internet2.edu/earlyadopters/draft-internet2-ea-mw-business-case-00.pdf for a thorough discussion of the values provided by enterprise directory services.

enterprise directory infrastructure

The infrastructure required to support and maintain an enterprise directory. This may include multiple directory hardware components as well as the processes by which data flows into and out of the directory service.

ETL

Tools handling data extraction, transformation, and loading are called ETL tools. These tools are common in the data warehousing industry, but are not yet commonplace in the directory/identity management industry. While directory industry metadirectory products and directory integration products such as IBM Directory Integrator (formerly *Metamerge Integrator*) may be able to address this need, most institutions make use of Perl scripting.

GUID

GUID stands for Globally Unique Identifier. A guid is a unique identifier intended to identify a single person for the entire period of their intersection with an institution's electronic services. The guid is intended to function as the primary key for an individual within an institution's enterprise directory, serving as a permanent link between all identifiers for an individual. Guid's may be assigned according to an algorithm or constructed from source system identifiers. Guid's should not be changed, reassigned, or retired. (The term "global" in this context means "within an institution", not across all institutions.) [See Addendum 4.5.2 – *Why Social Security Number (or Any Government identifier) Is NOT A Good Guid* and Early Harvest: Identifiers, Authentication, and Directories: Best Practices for Higher Education <http://middleware.internet2.edu/internet2-mi-best-practices-00.html> for a thorough discussion of related issues.]

intelligence

Intelligence is the processes that move data: a) from source/owner systems to the registry; and b) from the registry to one or more consumer systems (presumably one of which is a directory server) in conjunction with applying logic during the movement of the data, *e.g.* identity management, business rules, application of data formatting standards, *etc*.

Join

The Join is the process where disparate identifiers for multiple source systems are extracted and examined, thereby producing a single master record of identifiers for each individual entity which can be used as a link back to the source system records.

Kerberos

Kerberos is a network authentication system for use on physically insecure networks, based on the key distribution model presented by Needham and Schroeder. It allows entities communicating over networks to prove their identity to each other while preventing eavesdropping or replay attacks. It also provides for data stream integrity (detection of modification) and secrecy (preventing unauthorized reading) using cryptography systems such as DES.

LDAP directory

An LDAP directory is one that supports the Lightweight Directory Access Protocol (LDAP). LDAP is a widely adopted IETF standard directory access protocol well suited to the authentication and authorization needs of modern application architectures. SunONE Directory Server (formerly iPlanet Directory Server), Netscape Directory Server, OpenLDAP, Novell eDirectory, Oracle Internet Directory, and Microsoft Active Directory are examples of LDAP directories. (For more on this topic, see RFC1487 http://www.ietf.org/rfc/rfc1487.txt, RFC1777 http://www.ietf.org/rfc/rfc1777.txt, RFC2251 http://www.ietf.org/rfc/rfc2251.txt, and the LDAP Roadmap http://www.kingsmountain.com/ldapRoadmap.shtml.)

metadirectory (also ***meta-directory***)

The metadirectory represents processes where source data is captured, transformed, and presented in an enterprise directory.

RDBMS

RDBMS stands for Relational Database Management System. Examples of relational databases include Oracle, Microsoft SQL Server, Sybase, and Red Brick.

registry

The registry is the system in which identity of resources is resolved. This often refers to a database component of the enterprise directory.

alt. def.: Data taken from multiple source/owner systems to which "intelligence" has been applied in preparation for feed to one or more directories, applications, or other consumer systems. Further "intelligence" may be applied as part of any individual feeding process. Registry data may be housed in a relational database, indexed files, or a directory server.

Usage Examples

The following usage examples describe actual implementations using the above terminology.

I. University of Maryland, Baltimore County

The UMBC source data systems are the human resource (HR) and student information systems (SIS) with data stored in an Oracle RDBMS. Database triggers create updates in a change log. The updates are then applied to records in an iPlanet Directory Server LDAP v3 server. Perl scripts query the iDS change log for updates and accordingly update Microsoft Active Directory, the Remedy trouble ticket system, and NIS.

In this example, the collection of Perl scripts and database triggers comprise the intelligence function (presumably there is more going on here than merely moving data around). The iPlanet directory server functions as the registry.

II. University of Alabama, Birmingham

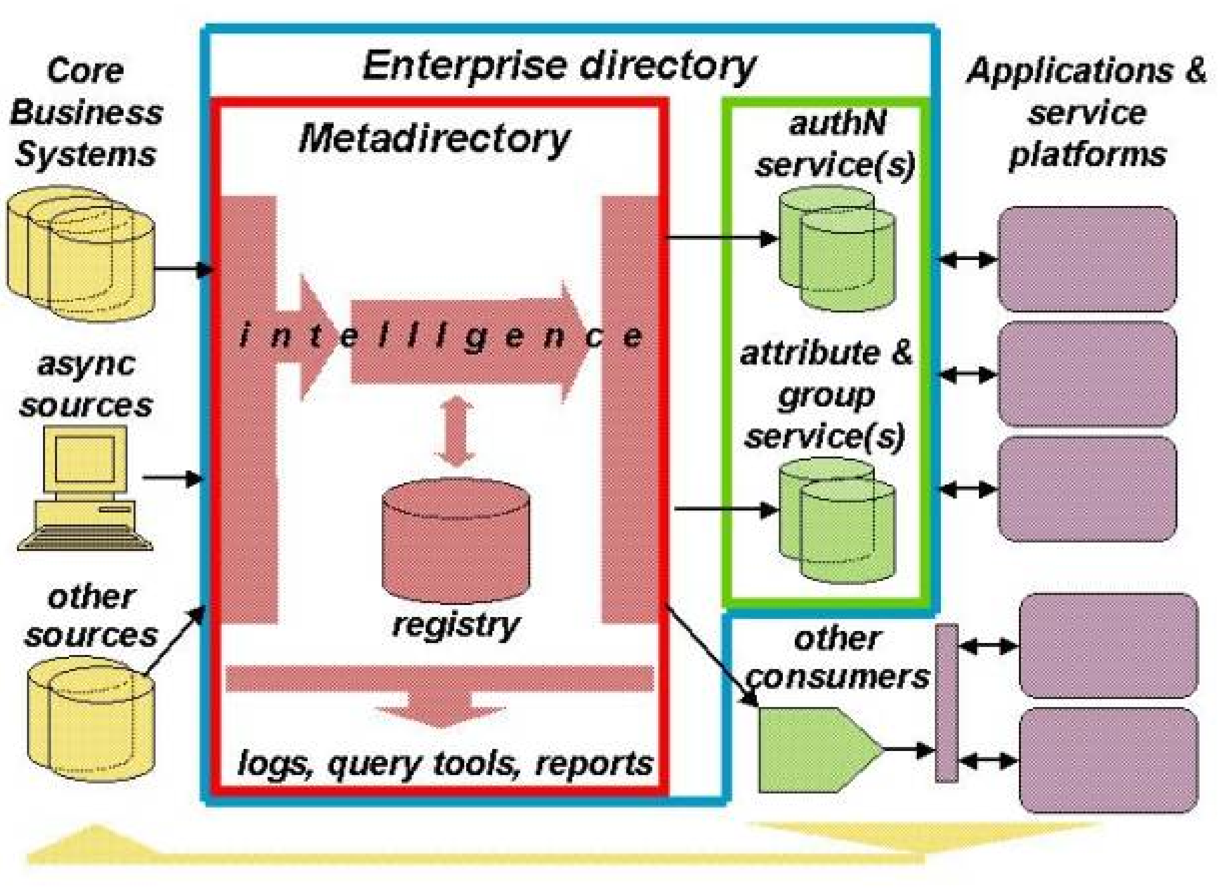
Source systems for UAB include the student system, HR, and a private Health system. Mainframe programs generate extracts which are then dumped to a qi/ph/CSO Nameserver server. Data is then pushed from the qi/ph server to an iPlanet Directory Server LDAP v3 server, and from that location to Microsoft Active Directory. The intelligence function is comprised of the mainframe programs which generating the extracts and the scripts updating qi/ph, iDS, and Active Directory. The qi/ph server effectively functions as the registry in this example.

III. Boston College

The BC registry **is** the source data system for identifiers and the single point of entry for all systems, including PeopleSoft HR. All new BC users are added first to the "corporate database" (VSAM files). This corporate database effectively functions as the registry. A set of unique identifiers is generated for use by all systems, obviating the need for identity reconciliation. Student and/or HR systems "activate" the user which marks the user for inclusion in the multiple feeder processes populating iPlanet Directory Server LDAP v3 server and other consumer systems (email, voicemail, NT, Radius). Certain transactions trigger near-realtime (asynchronous) updates whereas others are applied in batch nightly. In most cases, updates are shipped to the consumer systems via FTP and applied by scripts and/or C programs run on those systems. Intelligence is comprised of programs (batch or online transactions) that enter a user onto the system initially, the triggered near-realtime and bulk update routines, and the various programs that feed the data to consumer systems.

4 Planning for the Enterprise Directory

4.1 Metadirectory Processes



This model consists of three major processes. The first process is consolidating data from all systems of record, such as human resources information systems, student information systems, email address tables, UNIX account information, campus telephone directories, physical office locations, etc. All information is then "joined" to produce a single master record for each individual. This identity matching process resolves records that appear to be related to an individual, determining definitively whether they are or are not. The resultant collection of resolved master records, referred to as a "registry", may be stored in a single data store (database table, indexed file, *etc.).* In essence, this process reviews all of the relevant institutional sources of data and joins them together.

The second process, termed in this document "intelligence", manages how data is inserted, modified, and deleted from the registry based upon the business rules of the institution. This process is mindful of both the data providing source systems and the applications that will consume the transformed data.

The third process considers all the applications and systems using this enterprise infrastructure ---the consumers of directory information --- and provisions them accordingly. For example, directory-enabled applications such as calendaring may require an LDAP directory presentation of the data. Non-directory-enabled applications may require special presentations, perhaps of just a few of the attributes. Resource provisioning and account management systems track additions, removals and changes of status and perform tasks accordingly.

With implementation and acceptance of these processes, and confidence in the enterprise directory, a metadirectory process can be leveraged to update core business systems. The ownership of certain data associated with the system of record may migrate to the enterprise directory. Data owned by the enterprise directory may be pushed back into the systems providing data to the enterprise directory. These issues are just beginning to be addressed at many campuses. Many campuses have yet to reach this stage of enterprise directory development.

It should be noted that this document considers issues primarily relating to information about people, the most common starting place when building an enterprise directory infrastructure. This is also where the most experience is, and where best practices have been most clearly defined and published. This document does not consider information such as organizational entities, physical information, computer and network nodes, or other types of resource data stored in enterprise directories. The implementation described in this document is intentionally "vanilla" and represents the most common practice, given current experience.

4.2 The Join: Directory Sources, Identity Matching, and Registry Building

The Join process copies data from institutional information sources, creates a resolved entry for individuals, and moves it into a registry to service application requests. Because institutions often have multiple systems of entry for people information, and identity policies may not be consistent across all systems, the main difficulty is reconciling records from the multiple sources into a single record for each person.

Some institutions have enacted policies and processes to minimize the need for identity resolution through a join process by providing a single institution identifier regardless of the type of affiliation. This proves useful in environments where people may have multiple simultaneous affiliations such as student and staff, as well as migrate among affiliations over time such as undergraduate student and graduate student. This requires a central system to supply system-wide identifiers and identity management policies. Such policies should be applied to all systems creating people records. A system-wide identifier can then be used to link all systems containing people information.

4.2.1 Directory Sources

Institutions maintain authoritative sources for their data. Two common and important authoritative sources for people data are the human resource (HR) and student information systems (SIS). The HR system may provide faculty and staff information including name, job title, a tax identifier or nationally assigned identifier (such as the U.S. Social Security Number, for which the U.S. Social Security Administration provides regulations as to its use), and home address, usually keyed by employee number. The student information system (whether home-grown, PeopleSoft, SIS, SCT Banner, or something else) generally contains student information such as name, class level, major, and permanent home address, usually keyed by a student ID.

Any system that masters information about people may be a valid source for directory information. This includes systems which track alumni, vendors, donors, parents, and other affiliates. Campus housing, facilities management, telephone, and email systems may also provide important contact information useful to include in an enterprise directory.

There may be other sources that are updated occasionally, such as those holding data about campus contractors or loosely affiliated agencies. The library system may supply information about general public patrons, and the medical center may provide data about affiliated doctors or even patients. To provide services such as computing access, recreation center access or meals to summer conference participants and campus guests, administrators may supply information about them as well.

Some directory information may be updated asynchronously, *i.e.*, in near real-time. Information from the types of sources mentioned in the preceding three paragraphs might be provided in batch extracts as well as be available via transactions. The registry or other parts of the enterprise directory infrastructure themselves authoritatively house certain directory information. Applications updating that information may serve as asynchronous data providers.

The physical implementation of directory sources can include application databases, indexed files, flat-files, operating system directories, and web sites. Any system storing information about people affiliated with your institution should be considered a possible directory source.

4.2.1.1 Issues with Directory Sources

Identifying the authoritative campus source for each attribute is a critical component in designing the enterprise directory infrastructure. One of the biggest challenges reported by institutions that have built these infrastructures is maintaining the integrity (or correctness) of the data in the enterprise directory as some portion of the data in their source systems is out-of-date, contains mistakes, and/or is not consistently formatted. Most institutions prefer not to fix bad data within the enterprise directory and instead develop a policy stipulating that corrections must be applied at the source. While such policies reduce the amount of transformations required to handle erroneous data, they may undermine the usability of the directory for consumers if the administrators of the source systems are less than responsive.

Another common problem is deciding which pieces of source data to use in the enterprise directory. For example, one institution surveyed allowed thirty-four distinct address types in their student information system and had to establish a separate project to determine which address type was the appropriate one to move to the directory. Additional source data examples include how to order multiple titles, so that the president's information is displayed primarily as "president" or "officer" rather than "faculty". These require applying knowledge of the business, policies, and the source systems (as well as in some cases allowances for personal preference), all of which are different at each institution.

One of the most valuable outcomes of building an enterprise directory infrastructure is simplifying attributes. For example, determining which address is the appropriate one, adding it to the directory and supplying it to all applications that need it, is a great time saver for developers and application integrators. In a similar fashion, computed values that reduce many attributes to a few are also useful. For example, several categories of faculty (associate professor, lecturer, *etc.*), staff (part-time, professional), or students (graduate, undergraduate) stored in the systems of records can be aggregated into attributes designed to reduce both the complexity and runtime of queries. Another example involves tracking student fees and evaluating whether the appropriate fees are paid in determining library access. While there may be many different kinds and amounts of student fees tracked in the student information system, the enterprise directory could provide a single attribute summarizing fee status for the library application to use.

It is common to transform data before putting values into a registry. Standardizing format, case in names, and attribute contents are the most common, along with removing duplicate names coming from different systems. Directory planners should keep in mind that data transformation can require a significant investment in time and energy.

4.2.2 Identity Matching

The most challenging problem is matching up a person’s identities and deciding when records from different sources apply to the same or different individuals. Typically, most overlap occurs between the HR and student systems where a staff member enrolls in a course or a student holds a part-time staff position. A common strategy is to compile a list of attributes and use them as a basis for comparison. Attributes that infrequently change are often used, such as U.S. Social Security Number, formal name, date of birth, and permanent home address. While every institution uses its own approach and criteria, the common goal is to automate as much of this as possible to avoid the potentially high costs and inefficiency of manually reviewing and resolving mismatches. David Wasley of the University of California system reports that their matching logic fails to resolve about 1000 out of 200,000 people, a success rate of 99.5%. Of course, the ability to code matching rules successfully depends on the quality of data you are trying to match. Systems for which data is not entered with care or for which it is acceptable to falsify or leave absent data, will have greater problems trying to reconcile identity.

One variation on the strategy of exact-matching tries to address the problem of non-synchronized identity attributes by using a point-matching scheme where each possible attribute match adds points to a cumulative matching total. If the total is greater than a pre-set calibrated number, the two records are considered to be a likely match. This approach or similar fuzzy-logic techniques may offer some hope for institutions that have not enforced consistent and complete data entry across their systems.

In general, the effectiveness of identity matching is controlled by the consistency, quality, and amount of data. The more information supplied from each source system, the more successful the coding of matching rules. It is harder to perform good matching with only name and address, easier with the addition of date of birth, gender, government-assigned id, and easiest if they all match.

It may also be important to note what steps were taken to verify the authenticity of identification information when people were entered into source systems. Government-issued identification, such as picture id's, U.S. Social Security Number cards or other national id's, birth certificates, and passports, often provide an adequate degree of verification, and often student and HR systems have policies to ensure proper identification. Other campus systems may allow access to services without strong identification verification. For example, the campus library may register patrons with little or no verification. It is particularly important to consider the strength of identity verification policies if the enterprise directory will be used to supply authentication services.

Once two or more records are matched and given a unique identifier, updates can be handled automatically when the source data changes. However, if different sources map to a single attribute in the registry, then programming logic must determine which source is the authoritative one. On one campus, for example, different sources contained different gender information for the same person. It is important that for any single attribute in a single registry entry there is a single system of record.

4.2.3 Building the Registry

Building the registry entails extracting, transforming, and loading (ETL) the data into the registry. Many institutions manage these processes with Perl scripts or Java applications. Commercial products may also be an attractive option to institutions starting now and those who wish to move away from scripted solutions.

The recommended method for storing and managing the registry data is to use a relational database. The size of the institution and amount of data to be stored in the registry are factors to consider. Registries can be "fat" or "thin", depending on how much data is put into the registry. If the source systems are capable of being accessed by a variety of applications (perhaps using LDAP or SQL) and are highly available themselves, then building a thin registry with just enough data to perform the identity resolution might make sense, since the applications or consumers can get the identity from the registry and other data from source datastores. Most campuses choose to build a fat registry in order to supply information to consumers. Fat registries simplify processes required to meet application and consumer requirements and can help to avoid issues regarding the technical or procedural inaccessibility of source systems. Having more information about entries in the directory can also facilitate the use of *dynamic groups* – groups that are based on entry attributes – to provide authorization services. [For a thorough analysis of directory groups please see MACE Best Practices for Directory Groups <http://middleware.internet2.edu/dir/groups/draft-internet2-mace-dir-groups-best-practices-01.html>.] For these reasons fat directories are more common than thin directories.

Recommendations to consider:

· Instituting a common entry point for people can mitigate the complications of the join function. This will require more modifications in data entry systems but can be justified by cleaner registry content. As more vendor products become directory-enabled a central entry point becomes more feasible.

· Model the data in both source systems and the registry to ensure that attributes in the registry are populated from the definitive owning source and attributes are not over-loaded.

· Use aggregate and summary attributes in the registry to support and simplify common queries.

· Understand the identification, entry, and verification policies and processes used on source entry systems at your institution

4.3 Intelligence

Some aspects of metadirectory operation are not specifically source or consumer oriented, but rather are concerned with architecture and the operation of the registry itself. These considerations reflect how the institution's business rules and policies are implemented in the metadirectory, hence the use of the term "intelligence". Steve Carmody of Brown University identifies this function as the "lizard brain" - a form of primitive intelligence expected to evolve to a more advanced level in the future.

4.3.1 Unique Identifiers

The registry database assigns a key or globally unique identifier (*guid*) for each entry. [See Addendum 4.5.2:*Why Social Security Number (or Any Government identifier) Is NOT A Good Guid* and Early Harvest: Identifiers, Authentication, and Directories: Best Practices for Higher Education <http://middleware.internet2.edu/internet2-mi-best-practices-00.html> for a thorough discussion of related issues.] This process should be carefully architected so that guid's are never reassigned or revoked, and satisfy the requirement of being unique within a broad scope. In most implementations, the registry creates and owns this guid, typically a long integer or alphanumeric string. Some campuses employ a unique identifier based on the internal ID's generated by their administrative system. The important thing is that the guid should be constructed so that it is least likely to have to be changed.

The persistence of the registry entry is also a critical design factor. In the simplest metadirectory model, source data is captured, transformed, and consumer systems are reloaded all at once. This effectively makes consumers into mirror images of the source systems. A registry need exist only transiently in this model. This approach is not common and does not scale as the need for additional and more complex or computed data increases. The standard approach is for registry entries to persist for at least as long as the person to which the entry is bound is to be provided services. It may also be the case that a registry entry is deactivated but never deleted so that some history remains. This can be used in the case that a person who has left returns, to aid in long-term auditing, or to help ensure the uniqueness of guid's and other identifiers. The registry can also be used to enforce the institutional policy of reusing UID's and logins, if one exists.

The registry's structure must be designed to support data flows from sources to consumers, and possibly to other destinations, including flows terminating in the registry itself. At minimum a registry entry must store the guid and all the source and consumer systems' identifiers for an individual's entry. This is so that new information arriving from a source system can be associated with the proper registry entry and be used to update the proper entries in consumer systems. (The co-locating of source system identifiers together in a registry entry is the outcome of performing the join operation discussed in the preceding section.) Keeping all binding information together in the registry enables a metadirectory design that permits all source and consumer systems to maintain their "native" fundamental keys or identifiers. It also does not require the external systems to be extended to include the registry's guid for binding purposes.

If the guid is created in the registry, the initial design might specify that as the only place it will appear, and that it is only used internally to guarantee uniqueness inside the registry. Experience has shown that new consumers of directory information will request or even demand to include the guid in their representation of metadirectory information or application development. It is best to plan for this possibility from the beginning.

Allowing the guid to be used in applications may seem harmless, however, it can create problems. One problem is: in the event that metadirectory needs require that the format of the guid be modified, such modification will be complicated by the use of the guid in application databases. A second problem is, that application usage may create pressure to modify guid values, pressure which could not exist if the guid remained internal to the metadirectory. A third problem is that the release of the guid to application systems means that specific guid values and who they are assigned to may well become known to a larger group of people than just metadirectory administrators. This is problematic because guids are assigned for the lifetime of electronic service, and could become a problem in the event that a person requests confidentiality under the U.S. FERPA law (the Family Education Rights & Privacy Act, see http://www.ed.gov/offices/OM/fpco/ferpa/erparegs.html) or other privacy/confidentiality frameworks. Periodically each U.S. campus conducts reviews to determine what is or is not included in "directory information" as defined by FERPA and metadirectory administrators should work with their campus regarding these definitions.

One alternative to allowing the guid to proliferate is to promote the use of a different identifier for this purpose. One institution advocates that new system developers use a Publicly Visible Identifier (PVid) as a foreign person key. This, coupled with never releasing guid values, will reduce the chances that a person's guid will need to be changed over time. Whichever identifier is chosen, the registry bears a responsibility to maintain an accessible historical change log. Identifier changes can rarely be completely eliminated. If there are loosely connected systems using an identifier that changes in the registry, they will need such an historical change table to update their systems when they notice that the old value is no longer in use in the registry. This underscores the fact that whichever identifier is used, it should not be reassigned, or identity mapping between the registry and the connected systems can become corrupted.

4.3.2 Technical Expression of Institutional Policy and Procedures

The registry is the one place in which data is stored in a manner independent of source or consumer system constraints. Source data is transformed when it is added, and consumer data is derived when it is used. For example, standardized telephone numbers, addresses, and names might be stored in the registry once, so that one does not need data tailored for each consumer. This can require storing each discrete part of an address separately so that the complete address can be reformulated properly into the formats required by different consumer systems.

The registry must also contain information needed to support the reconciliation of data from disparate source systems, if one or more of those systems are not available during the registry-update processing. As an example, a business rule might mandate that a person's name is taken from the HR system, if the individual also has an entry in the Student Information System. However, when the HR system is unavailable, data sufficient to complete the entry must be kept in the registry to offset this. Two solutions might be: (1) maintain an attribute that indicates the registry's source of the name value or (2) maintain attributes that are copies of the values in the different source systems (SIS-name and HR-name, for example).

More sophisticated metadirectory designs may also address resource provisioning. Registry entries will need to contain data necessary to implement provisioning business rules. System accounts, email properties, group memberships, and values of certain attributes may all need to be controlled by business rules governing the provisioning policy. Extracts of registry information may also need to be prepared for export to non-live consumer systems within the provisioning umbrella. In addition, notification of the state or impending change of state of resources may need to be sent to account holders or others. Attributes describing the "state" of an entry and its associated resources and attributes containing the independent variables needed to evaluate provisioning rules, such as major affiliation and time at which the entry assumed its present state, will need to be incorporated into the registry. An example of stateful provisioning as implemented at the University of Memphis is detailed as an addendum.

4.3.3 Operational Design Requirements

Several factors affect the design of the information flow from the registry to the consumers. In the simplest case, consumers are updated by rebuilding them from scratch. In more sophisticated instances, consumers are rarely rebuilt, allowing updates to flow into them. In these instances, several design options should be considered. Should data be pushed, pulled, or should there be a publish-subscribe architecture? Should updates flow asynchronously (as they occur in real-time) or only in pre-scheduled batches?

Closely related to consumer data flow is the capability of the metadirectory to be used to recover from accidents. Systems and live-ware (*i.e.*, people) will fail to operate perfectly from time to time, and the metadirectory may blithely move bad data into consumer holdings, causing a corresponding impact to services from the users' perspective. The potential for a spectacularly grand failure exists. The risk for such an occurrence must be minimized. A batch flow model could afford a metadirectory operator the chance to review proposed updates before they are released. For example, a threshold could be established so that batch updates are permitted to proceed automatically if the number of changes in the update is less than the configured threshold, otherwise an operator could be notified so that manual review could be performed. In asynchronous flow designs, a stateful approach to provisioning can help to smooth over such episodes. This might be accomplished by designing the finite state machine that governs automated provisioning so that transition to states in which users' service is reduced or eliminated can occur only after a time threshold or grace period is exceeded. That grace period provides a window of time in which to recover from the failure without direct user impact. An even more robust solution would be a design in which all changes to the registry are serialized, a registry changelog is maintained, and these are used to support rollback and replay operations for consumer updating.

The metadirectory infrastructure is yet another in the suite of technologies operated by central IT. Operational needs, both ordinary and peculiar to this technology, must be anticipated. Activity and exception logging at strategic points in the flow of data into, through, and out of the metadirectory is crucial to backtracking some types of service issues (was the user's access denied because of policy executed correctly by the metadirectory, or did something misfire?). Helpdesk staff will need general information on metadirectory status in order to anticipate potential changes to integrated services. Additionally, they will need a tool with which to query details concerning individual users to enable them to help diagnose specific service issues. There may be needs for standard or ad hoc reporting. Standard reports might provide detailed reporting of changes to certain subpopulations of interest (which faculty accounts are in transition to a disabled state?), circulate exceptions to appropriate operational staff, and outline how many of which types of changes to the registry occurred over the previous 24 hours. Ad hoc reporting needs can sometimes be met by relying on a metadirectory consumer's database, but it may prove most convenient to report from the registry itself for some purposes. In fact, that may be an early indicator of a new consumer needing to become integrated with the metadirectory. For example, certain of the information for printing the campus telephone directory may reside in the registry, and it may become only a matter of semantics whether you call the corresponding report a report or a consumer flow of occasional use.

There is one further requirement for metadirectory logging that is peculiar to this technology. The primary function of the metadirectory is to manage the relationship between all of the key identifiers that are associated with people in various databases maintained by the enterprise. Changes to this relationship must be audited. That is, logs must be kept when someone is issued a new identifier or re-issued an old one, or if an identifier is re-assigned from one person to another (your design should likely prohibit this, but it may still be possible to occur). It must always be possible to determine who had which identifier at which point in time, for all time. This information is needed to help maintain the integrity of some consumer systems, and may be needed to maintain the integrity of the registry itself. This information is required to be able to audit users' actions across the suite of services provided through consumers integrated with the metadirectory.

Recommendations to consider:

· The registry id, or guid, SHOULD NOT be reused or changed over time. It is less likely to require changes if: it is sized sufficiently for growth; is not constructed from id's that are likely to change, such as surname or department; has no embedded meaning; is not a vendor-controlled or otherwise externally-assigned attribute; and is not provided to applications.

· A publicly visible identifier (PVid) MAY be used as a foreign key to the guid for applications to use to prevent the need of publishing or propagating the guid.

· Maintain a history log of identifier changes.

· Understand and plan for the requirements of a provisioning system, defining needed attributes in the registry.

· Plan for errors in metadirectory processes and use operational thresholds and reporting to minimize risks and impacts of errors.

4.4 Consumers of directory information

This section describes consumers of the enterprise directory data and the processes that present the data to them from the registry. It is forward looking to applications that are either directory-enabled now or will be in the future. It is the area that primarily justifies building an enterprise directory infrastructure and is the payoff for that investment. Because this is the area of most active continuing development in the metadirectory infrastructure, it is the least settled and structured. This section covers the most common, vanilla, consumer functions that have appeared to date.

It should be noted that while earlier sections have discussed the registry concept without specific focus on implemented technologies, this section is focused specifically on LDAP, which is at the time of this writing the clear directory technology of choice. If an application is said to be "directory-enabled" it means that the application is capable of interfacing with an LDAP directory for at least authentication and possibly as a data store as well.

4.4.1 Supplying Multiple Consumers

The most common presentation of the data in the registry is an LDAP directory which is often used initially for white pages lookups and later for other directory-enabled applications. Previously, it was commonly thought that in building a white pages directory, the institution was also building the enterprise directory infrastructure. Now it is seen as only one consumer of enterprise directory information, typically the first to be deployed, and usually the one used to justify the initial investment. People information is most often the starting point for both the registry and the LDAP directory because it has immediate applications such as web searches and underpins other kinds of data. The success of deploying an LDAP directory with people information will naturally give rise to demands for other related kinds of data in the registry.

Why isn't a single LDAP directory presenting the data in the registry enough? As directory-enabled applications have come along, questions of whether to extend the schema of a master directory or build a special purpose directory tailored to the application have arisen. No one answer fits all situations, but most agree that some extensions are appropriate for the master directory and others work best with a special purpose version. Questions of what the application wants to store in the directory play into this. Directories are a natural place to store user preferences, so adding schema extensions to hold user preferences for the interface of a calendaring application would usually be seen as appropriate, but to put the actual calendar data in the directory would not. Most calendar products follow this division. Data that is frequently updated and/or application-specific is probably best stored in an application specific directory or application database, rather than an enterprise directory.

Similarly, depending on the design of directories, a special purpose version might be needed to hold different access permissions (ACL's) -- perhaps to allow users to update fields, while the master directory remains "read-only."

Directory replication strategies are often implemented to address performance requirements. How directory data is replicated influences directory designs, both in terms of what campus information is to be loaded into the replicants and the replication features of the particular LDAP software used. It may also make sense to have a directory replicant outside of the campus firewall that contains only publicly accessible contact information,. That way even if the replicant is maliciously hacked, private contact information would not be compromised.

It is important to make certain that private information is not propagated into replicant directories, application directories, or network operating system directories without also providing the necessary safeguards to prevent the information from being accessible to inappropriate persons. The campus policies and procedures used to protect private data in database systems may need to be extended to address directories as well. Campus data stewards/owners should participate in the review of projects involving consumers of registry data.

4.4.2 Authentication and Authorization

How the campus handles authentication (*authN*) can dictate choices in directory deployment. Although an LDAP directory can store the user password and thus be used to authenticate against, many institutions choose to build on technologies designed specifically for authentication, such as Kerberos. Since many applications may be directory-enabled but not Kerberos-enabled, it is convenient to use an LDAP directory authentication plug-in to pass authentication credentials to a Kerberos domain rather than storing them in the directory. In that way users can authenticate against the LDAP directory or the Kerberos domain and in both cases the Kerberos server validates the credentials. Several schools, including the University of Notre Dame and Georgetown University, have written such plug-ins, and work is currently being done in coordination with Sun to develop a more generic plug-in to be offered freely to higher education iPlanet Directory Server users. Other ways to authenticate, such as the WebISO project, also make an independent authN database, such as Kerberos, more attractive. Indeed, from the metadirectory infrastructure, the Kerberos realm can be viewed as just another consumer of metadirectory information (the Kerberos principal attribute, if not the password).

Authorization (*authZ*) is usually initially thought of in terms of LDAP directory groups and roles, although it is broader than that and includes elective provisioning discussed below. For an excellent discussion of directory groups as well as good advice, see MACE Best Practices for Directory Groups <http://middleware.internet2.edu/dir/groups/draft-internet2-mace-dir-groups-best-practices-01.html>. "All faculty," "all students" and similar groups can be created via metadirectory processes once the more or less arduous work of agreeing on business rule-based definitions for them has been completed. Such coarse-grained affiliation groups will likely find use in a broad range of consumer systems for authorization decisions and other purposes.

4.4.3 Support for Microsoft Active Directory

If Microsoft Active Directory is present on your campus but does not have the role of primary enterprise directory and integrating AD into your enterprise directory infrastructure is desired, then the following advice is offered for consideration.

In many cases, the initial load of Active Directory will come from an upgrade process where NT user accounts are migrated to AD. Unlike feeding AD from scratch from the registry, this allows sites to preserve a user's SID history and liberates NT administrators from having to re-ACL permissions on shares in conjunction with the migration. An initial process to perform reconciliation with AD user entries and registry entries will need to occur. This should be a one-time effort after which future adds/deletes/modifies would be fed from the registry. One way of performing this initial reconciliation process would be to produce LDIF (LDAP Data Interchange Format) output of both the user registry and the Active Directory and write scripts to identify discrepancies and apply updates as needed. Many directory integration products provide Active Directory connectors, so the use of a tool may be an attractive alternative to scripted solutions.

One technique worth considering for assuring reliable mapping between registry entries and AD entries is to track the Active Directory GUID (Global Unique Identifier) in the registry. Active Directory generates a GUID for all objects in the directory, including user objects, at the point the object is created in the directory. The GUID is guaranteed to be unique and is never changed, even if the userid or DN is changed (see previous discussion of GUID). Tracking this field would require additional scripting to extract the GUID from AD at the point a user is created.

In keeping with the recommendation that Active Directory be treated as another consumer system, campuses should consider loading only minimal people information to the Active Directory, e.g. common name, uid, etc. Bear in mind that Windows 2000 desktop systems will query the Active Directory by default for "Find People" lookups. FERPA regulations and local privacy policies should be kept in mind when deciding what data about users should be included in AD. It is possible and may make sense to set domain registry policies to direct standard LDAP searches such as the Start menu "Find People" option to your enterprise LDAP directory (see Addendum 4.5.3 for a means of accomplishing this).

A major issue with AD integration is what to do about passwords -- how to connect passwords in the LDAP directory or Kerberos database with the AD password. Some campuses use their Kerberos realm to control logging into Active Directory, using Microsoft's passthrough authentication capability, and set a random password in AD that is unknown to the person, and unused. This works only if there are no down-level clients to be supported. Other campuses develop strategies to synchronize the password in AD and the principal password, whether in Kerberos or another LDAP directory, by adding code to the agent used when a user changes a password.

The MIT Project Pismere site http://web.mit.edu/pismere contains additional technical information and documentation on Microsoft Active Directory issues.

4.4.4 Resource Provisioning

Resource provisioning is the automated handling of the tasks associated with the establishment, modification, and deletion of resources and entitlements provided to people as they join or leave an organization or undergo changes in affiliation or status. Since most institutions have already built pieces of this kind of infrastructure, higher education institutions may choose to extend and improve their existing processes by leveraging the enterprise directory information, rather than considering commercial provisioning products. Indeed, the Burton Group predicts a convergence of resource provisioning products and general metadirectory products. For campuses, the challenge is to directory-enable existing procedures, such as creating email accounts following the creation or update of a directory entry by the student information system.

Campuses considering these resource provisioning functions are addressing the problems of how to code and use the campus business rules. That is, when a person becomes a "student," what services should he/she receive, including computer accounts, disk quota sizes, home page location, *etc*., and how do the various services get provisioned? Institutions that have already addressed this have typically relied upon Perl scripts, with some campuses looking at directory integration products.

Architecturally, there are some common functions that need to be implemented for various provisioning tasks: create user, delete user, change name, enable/disable account. One key element is correctly identifying the consumer entry you are dealing with, since the consumer directory may not store the registry guid. More generally, how do you accommodate binding entries between the metadirectory (or registry) and consumers to insure you are dealing with the same person in both?

Next is what to use to code the business rules. While it has been suggested that XML is now the method of choice, no one is known to have an XML-based provisioning system in production at the time of this writing. The IBM Directory

Integrator product allows business rule logic to be contained in the scripting of assembly line connectors. This is in contrast to aggregating the rules and policy in one place so various agents can use them.

In higher education, many services are elective as based on status. Thus faculty and staff may be eligible to use a campus calendaring system, but not required to. Directory information such as status must therefore be presented to a consumer. For example, at the time a web application is used to activate an end-user's calendar, the necessary schema-extension attributes must be added to the user's directory entry and perhaps copied into the registry.

Recommendations to consider:

· While white-pages applications may be the initial purpose of an enterprise directory, it is rarely the end. Plan ahead.

· Develop a policy for determining when extending the enterprise directory schema is appropriate and when it is not. Be aware that application developers may want to store attributes in the enterprise directory inappropriately at times, as well as store attributes in an application directory that should be in the enterprise directory. Policies developed beforehand with the approval of the CIO or IT-head can help to navigate these issues.

· With the growth of the enterprise directory it will become necessary to copy directory information into replicants, application directories, and operating system directories. Work with campus data stewards/owners to institute policies and processes to protect the privacy/confidentiality rights of campus members.

· While LDAP directories can be used for authentication, consider the use of authentication technologies such as MIT's Kerberos. Authentication plug-in's to allow authentication credentials to be passed from an LDAP directory to a Kerberos domain MAY be used to support non-Kerberos-enabled applications that are directory-enabled. (Verify that the DSA of your choice supports authentication plug-ins.)

· Understand and plan for the requirements of a provisioning system.

4.5 Addenda

The following addenda are provided as aids and are not necessarily examples of "best practices".

4.5.1 University of Memphis Finite State Machine Provisioning Model

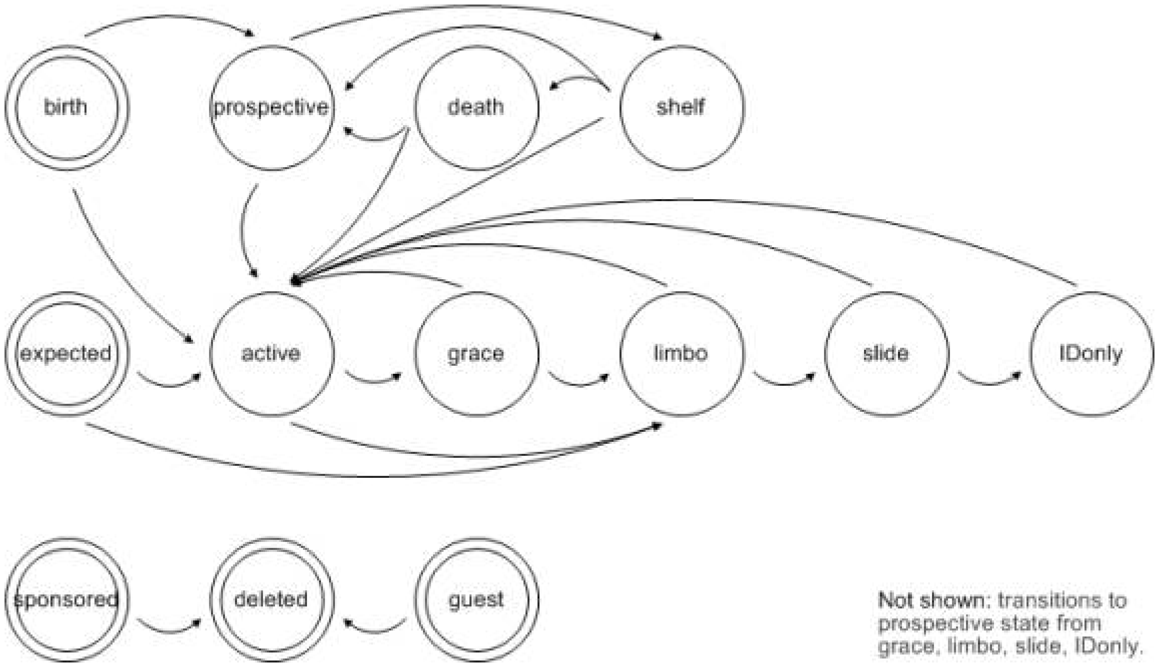
Basic account provisioning is guided by the finite state machine depicted below. It is designed to support three basic levels of service:

· Full access to basic services. For faculty, staff, and enrolled student affiliations.

· Email & identity management, including PIN maintenance for access to administrative web applications. For accepted student & registered student affiliations.

Identifiers maintained for continued support for outsourced services. For alum affiliations.

The model relies on the independent following data: state, substate, date the present state was reached, date by which the present state might end (called the expiration date), major affiliation (faculty/staff or student), and a multivalued attribute holding the identifiers of resources being managed for this account. Managed resources include shell accounts, IMAP/POP/HTTP mailbox service, campus-wide computing cluster access, and appearance in several directories used by a variety of application and web services for authentication, access control, or to determine eligibility for service.



4.5.1.1 Descriptions of states

Birth. This is a special designation that exists transiently while a registry entry is being initially created. It can occur during a merge process that is processing an SIS, HRS, or APR feed, whether that feed is batch or asynchronous. Basically, birth is the state that initiates all automatic account creations. All manual account creations are handled as *expected*, *sponsored* or *guest*.

Expected. A state entered manually which is provisioned exactly as the *active* state. Only persons asserted to have either *enrolled student*, *staff*, or *faculty* affiliation with UoM should be given an *expected* state. The *expected* state differs from *active* in that the account is given an *expire_entry* attribute containing a date by which they must appear in a feed from a core business system in a way that demonstrates their having one of the above affiliations.

Active. The typical state of an account of a member of the community in good standing, i.e., someone with either *enrolled student*, *staff*, or *faculty* affiliation.

Grace. A period of 180 days after all *enrolled student*, *staff*, or *faculty* affiliation ceases during which all resources and access persist. At three points during this state the person is emailed about their impending loss of access. Currently the third digit of *qistate* is used to track how many notifications have been sent and so when another is needed. We will add a new attribute named *uMemphisGraceWarnings* for this purpose.

Limbo. A 30 day period in which nothing changes. Transition into *limbo* is accomplished by disabling all access, but leaving all resources intact. This is a failsafe against deleting the resources of someone who believes they still need service, or who has been an exception of some type.

Slide. A variable length period during which resources are deleted. *Slide* ends upon notification reaching the metadirectory of the deletion of the last resource. Among the resources deleted are all but a specified set of attributes appearing in consumer directories.

IDonly. All "identity related information" will be maintained in the registry for all persons who have ever had an account in the *active* or *expected* states. They are deleted from metadirectory consumers. Identity related information will include, at least, *universal_userID*, *name*, *surname*, *givenname*, *birthday*, *id*, *uMemphisSSN*, *password*, *email*, and any identifiers remaining that bound this entry together with entries in source systems, such as *plusID* and *aprID*. No actual service can be rendered to an account in this state. Instead, it is useful only as a repository of usernames (and other identity data) which we can send to vendors supplying outsourced services. Persons with *alum* affiliation will typically have an account in this state, at least until such time as we decide to provide them some service such as email redirection. We will need to preserve *universal_userIDs* indefinitely before we can completely implement this state.

Prospective. This is the typical state of an account for someone with either *accepted student* or *registered student* affiliations, and who doesn't also have *enrolled student*, *staff*, or *faculty* affiliation. The account should be provisioned so that it can be used on http://iAM.memphis.edu and for email service.

Shelf. Similar to *IDonly*, but with *expire_entry* set so that an account remains in this state for 2 years. This state exists so that a *universal_userID* given out to someone for whom we won't keep it for life can be reassigned, but not for at least two years to make interpretation of system audit logs easier for that period of time.

Death. Similar to *IDonly*, but *universal_userID* and any source system binding identifiers (such as *plusID* and *aprID*) are removed. Accounts entering *death* have their *universal_userIDs* made available for reassignment. The name of this state is unrelated to the condition of the person to whom it is assigned!

Sponsored. Similar to active, but outside the control of the automated account provisioning system.

Guest. Accounts built by the manageGuest facility only exist in the ldap directory and are not managed by the state machine. These have a short lifetime after which they are automatically deleted. They only provide ability to authenticate using the LDAP BIND operation, and so enable official guests to use our dialup, wireless, and wireline network access. The ldap *uid* attribute uses a namespace orthogonal to the *universal_userID* namespace.

Deleted. As it says, the account is deleted from wherever it appears.

4.5.1.2 Transition rules

Birth → Prospective. Receipt of a SIS record for a new person of *accepted student* or *registered student* status causes creation of an account in *prospective* state.

Prospective → Active. Occurs upon receipt of a SIS record indicating *enrolled student* affiliation, an HRS record indicating *staff* affiliation, or an APR record indicating *faculty* affiliation for the person.

Prospective → Shelf. Occurs upon loss of last qualifying affiliation, i.e., to have been in *prospective* state the person must have had a *umAffiliation* value of *accepted student* or *registered student* or both. When the last of these is removed transition occurs. The *expire_entry* field is set to 2 years hence.

Shelf → Death. Occurs upon occurrence of the *expire_entry* date. That is, the first time after that date that the state machine examines this record (perhaps during a merge process) certain fields are removed from the registry entry (as described above) and the corresponding entries in metadirectory consumers are deleted.

Note. Not shown are transitions from *limbo* or later states into *prospective* state. In effect, they'll stay on the *limbo → slide → IDonly* track, but their *umAffiliation* may show *accepted student* or *registered student*, which will provide them with authorization equivalent to the *prospective* state.

Birth → Active. Receipt of a SIS, HRS, or APR record for a new person of *faculty, staff,* or *enrolled student* status causes creation of an account in *active* state.

Expected → Active. Occurs upon receipt of a SIS, HRS, or APR record for a new person of *faculty, staff,* or *enrolled student*.

Expected → Limbo. Occurs if the present date is greater than or equal to the registry entry's *expire_entry* field. Uses fast track removal. The *expire_entry* field is reset to 30 days hence.

Active → Grace. Occurs upon loss of last affiliation qualifying for *active* state, i.e., to have been in *active* state the person must have had a *umAffiliation* value(s) of *faculty, staff,* or *enrolled student*. When the last of these is removed transition occurs.

Active → Limbo. Occurs upon invoking fast track removal. The *expire_entry* field is set to 30 days hence.

Grace → Limbo. Occurs if the present date is greater than or equal to the registry entry's *expire_entry* field, or if fast track removal is invoked. The *expire_entry* field is set to 30 days hence.

Limbo → Slide. Occurs if the present date is greater than or equal to the registry entry's *expire_entry* field. The *expire_entry* field is removed.

Slide → IDonly. Occurs when the *accounts* field is empty, i.e., after the last resource managed by the state machine has been deleted.

Whatever → Active. Occurs when the person shows up in a source feed with either *faculty*, *staff*, or *enrolled student* status. The *expire_entry* field is removed if present.

Whatever → Prospective. Occurs when the person shows up in a source feed with either *accepted student* or *enrolled student* status. The *expire_entry* field is removed if present.

4.5.2 Why Social Security Number (or Any Government identifier) Is NOT A Good Guid

There are several characteristics that should be considered when selecting a Guid. While it is possible to shortcut the analysis by simply using an identifier provided by an external entity – say the local or national government – this is strongly recommended against.

For many years common practice in the United States has been to use the government assigned Social Security Number (SSN) as a primary identifier in information systems for people. The SSN seems tailor-made to serve as a primary identifier for several reasons: its uniqueness is assured by the government, the scope of its uniqueness is national, it is non-revocable and generally not reassignable, almost all relevant persons have one and only one, it has some degree of opacity (less than most believe as it seems to be but actually is not randomly assigned), it is short and easy to memorize and record, it contains only typeable and readable characters, it does not contain whitespace or special characters like quotation marks or symbols, and it can be verified by means of the SSN card provided by the Social Security Administration. What more could one possibly want in a primary identifier?

While it may be tempting to rely upon an external authority to solve your unique identifier problems, there are difficulties with this practice. In the United States the use of SSN as a primary identifier has come under fire for several reasons and legislation has been considered to significantly restrict its use.

One concern is that externally-assigned identifiers are outside of the control of your institution. An external identifier is essentially "owned" by another organization which can at will alter its format, function, or the rules by which it is populated and may do so without concern for the problems that will cause your institution. This reason is also valid for identifiers provided by vendor tools. A degree of independence is warranted to reduce risk to your identity system.

A second concern is that because nationally assigned identifiers are by definition extremely broad in their uniqueness (unique across all individuals who are citizens perhaps) there is an element of personal privacy and confidentiality that is put at risk by dependence upon it. Like a fingerprint, someone's national identifier may be very close to being their identity and as it is probably persistent for life its misuse, theft, or abuse can have disastrous and potentially fatal results on its owner. While your institution may have business rules in place to try to maintain confidentiality, because of the degree to which institutions of higher learning allow exceptions to policies it is simply impossible to ensure that confidentiality is never compromised.

A third concern is that not everyone who wishes to be affiliated with your institution may have a nationally assigned identifier. In these cases you have to generate an alternate identifier. If the person with such an institutionally-generated identifier is later granted a national identifier and then re-enters your institution's sphere of affiliation, processes may be required to map the national identifier to the institutionally-generated identifier or change the guid of the individual to the national identifier. This matching process can be complicated and problematic and begs the question: "if you are going to need a process for generating alternate identifiers anyway, why not design it to scale appropriately and always use it?"

For more information on identifiers and their characteristics and use, see Early Harvest: Identifiers, Authentication, and Directories: Best Practices for Higher Education <http://middleware.internet2.edu/internet2-mi-best-practices-00.html>.

4.5.3 Technique for Directing Windows Searches to an External LDAP Directory

The following code is used at the University of Notre Dame to set a Windows domain global policy to force the Windows Start button Find People searches to search the Notre Dame enterprise LDAP directory rather than the Windows Active Directory. The Account Name, LDAP Server, and LDAP Search Base would need to be set to those of the institution. Questions about this can be directed via email to the Notre Dame enterprise directory services team at eds@nd.edu.

```
Set WshShell = CreateObject("WScript.Shell")

ndEDS = "HKEY_CURRENT_USER\Software\Microsoft\Internet Account Manager\Accounts\Notre
Dame\"

WshShell.RegWrite ndEDS, ""

WshShell.RegWrite ndEDS & "Account Name", "Notre Dame", "REG_SZ"

WshShell.RegWrite ndEDS & "Connection Type", "00000003", "REG_DWORD"

WshShell.RegWrite ndEDS & "LDAP Server", "directory.nd.edu", "REG_SZ"

WshShell.RegWrite ndEDS & "LDAP Authentication", "00000000", "REG_DWORD"

WshShell.RegWrite ndEDS & "LDAP Server ID", "511", "REG_DWORD"

WshShell.RegWrite ndEDS & "LDAP Resolve Flag", "00000001", "REG_DWORD"

WshShell.RegWrite ndEDS & "LDAP Timeout", "60", "REG_DWORD"

WshShell.RegWrite ndEDS & "LDAP Search Return", "00000064", "REG_DWORD"

WshShell.RegWrite ndEDS & "LDAP Search Base", "o=" & chr(34) & "University of Notre Dame"
& chr(34) & ",st=Indiana,c=US", "REG_SZ"

WshShell.RegWrite ndEDS & "LDAP Port", "00000389", "REG_DWORD"

WshShell.RegWrite ndEDS & "LDAP Secure Connection", "00000000", "REG_DWORD"
WshShell.RegWrite ndEDS & "LDAP Simple Search", "00000000", "REG_DWORD"

WshShell.RegWrite ndEDS & "LDAP Bind DN", "00000000", "REG_DWORD"

WshShell.RegWrite ndEDS & "LDAP NTDS", "00000002", "REG_DWORD"

ndAD = "HKEY_CURRENT_USER\Software\Microsoft\Internet Account Manager\Accounts\Active
Directory GC\"

WshShell.RegWrite ndAD, ""

WshShell.RegWrite ndAD & "Account Name", "Active Directory", "REG_SZ"

WshShell.RegWrite ndAD & "Connection Type", "00000003", "REG_DWORD"

WshShell.RegWrite ndAD & "LDAP Server", "directory.nd.edu", "REG_SZ"

WshShell.RegWrite ndAD & "LDAP Authentication", "00000000", "REG_DWORD"

WshShell.RegWrite ndAD & "LDAP Server ID", "511", "REG_DWORD"

WshShell.RegWrite ndAD & "LDAP Resolve Flag", "00000001", "REG_DWORD"

WshShell.RegWrite ndAD & "LDAP Timeout", "60", "REG_DWORD"

WshShell.RegWrite ndAD & "LDAP Search Return", "00000064", "REG_DWORD"

WshShell.RegWrite ndAD & "LDAP Search Base", "o=" & chr(34) & "University of Notre Dame" &
chr(34) & ",st=Indiana,c=US", "REG_SZ"

WshShell.RegWrite ndAD & "LDAP Port", "00000389", "REG_DWORD"

WshShell.RegWrite ndAD & "LDAP Secure Connection", "00000000", "REG_DWORD"

WshShell.RegWrite ndAD & "LDAP Simple Search", "00000000", "REG_DWORD"

WshShell.RegWrite ndAD & "LDAP Bind DN", "00000000", "REG_DWORD"

WshShell.RegWrite ndAD & "LDAP NTDS", "00000002", "REG_DWORD"
```

5 Documents

Identifiers, Authentication, and Directories: Best Practices for Higher Education <http://middleware.internet2.edu/internet2-mi-best-practices-00.html>

MACE Best Practices for Directory Groups <http://middleware.internet2.edu/dir/groups/draft-internet2-mace-dir-groups-best-practices-01.html>

6 Advice To Implementers

Internet2 Middleware (MACE) website <http://middleware.internet2.edu/>

Internet2 MACE Authentication website <http://middleware.internet2.edu/core/authentication.html>

Internet2 MACE Authorization website <http://middleware.internet2.edu/core/authorization.html>

7 Change Log

This section lists the changes (other than typographical corrections) that have been made between released versions

April 2002 : Release for public review (RPR) of 1.0.

October 2002 : Release of 1.0 final.

1. Rewrite of the *University of Memphis Finite State Machine Provisioning Model* addendum.
2. Addition of the *Why Social Security Number (or Any Government identifier) Is NOT A Good Guid* addendum.
3. Addition of the *Technique for Directing Windows Searches to an External LDAP Directory* addendum.
4. Additional author and contact information provided.
5. Changes throughout the document in relation to the purchase of Metamerge by IBM.
6. Movement of the Enterprise Directory model diagram into section 4.
7. Addition of the *Change Log* section and renumbering of subsequent sections.

8 Acknowledgments

This document would not have been possible without the significant contributions of Richard Jones of the University of Colorado (retired) and fellow members of the MACE-Dir team (http://middleware.internet2.edu/dir/): Dr. Thomas Barton of the University of Memphis (also for the University of Memphis Finite State Machine Provisioning Model), Keith Hazelton of the University of Wisconsin, Michael Gettes of Georgetown University, Robert Banz of the University of Maryland, Baltimore County, Todd Piket of Michigan Technological University, Bob Talda of Cornell University, Ann West of EDUCAUSE/Internet2, and Renee Frost of Internet2.

The authors wish to acknowledge the contributions of Eileen Shepard of Boston College, and the editing assistance of Jeanette A. Fielden.

The authors also wish to thank the members of the NMI Integration Testbed and the MACE group for their valuable feedback.

The authors gratefully acknowledge the support of Internet2 and the National Science Foundation through the NSF Middleware Initiative (NMI) program.

9 References

Internet2 MACE Authentication website <http://middleware.internet2.edu/core/authentication.html>

Internet2 MACE Authorization website <http://middleware.internet2.edu/core/authorization.html>

10 Contact Information

Brendan Bellina, Editor
University of Notre Dame
Email: bbellina@nd.edu