Overview

The UA NetID (hereafter referred to as just NetID) is the name given to the common authentication registry for use by disparate, network applications within the University of Arizona computing environment. The following decisions were made during the design of the NetID service:

- The service must be based on standard protocols, available among popular computing platforms and application environments
- The service must implement robust security measures, both at the transport and data store levels
- The service should include a well-defined, not overly broad, functional scope. Functional overlap with other, pre-existing, application services should be avoided.

The NetID service addresses these goals as follows:

- LDAP v3 (RFC 2251) is used as the application protocol, and the NetID authentication registry is represented as a simple, flat directory structure. LDAP v3 is a nearly ubiquitous Internet protocol these days, comes bundled with many popular operating systems, and is accessible within nearly all popular web and client/server application development environments.
- Transport-layer security is accomplished by tunneling the LDAP protocol over SSL—commonly referred to as “LDAPS”. Data store security is facilitated via the use of access control instructions (ACI’s), which can be applied to directory branches and individual entries. Coupled with LDAP group entries, access to sensitive information can be restricted to appropriate sets of individuals.
- The set of attributes included in the NetID schema has been purposefully restricted to the bare minimum necessary to provide a useful authentication and identification service. The ability to perform authorization, link to external systems, etc, is outside the scope of the NetID service—and, as such, is deferred to more appropriate services, such as the Enterprise Directory Service (EDS).

This document will attempt to provide a high-level treatment of the NetID service, including:

- Inclusion and activity rules
- NetID directory schema
- Establishing a connection to the NetID directory
- Searching and retrieving information
- Object- and attribute-level access control rules
- Code examples
NetID Inclusion and Activity

Summary
At this point in time, students (as recorded in UAccess Student), employees, affiliates and associates (as recorded in UAccess Employee), and a small set of other constituents (primarily department sponsored visitors, or DSVs) are represented within the NetID authentication directory.

As of the Fall 2009 semester, all NetIDs are now "permanent"—that is, once a NetID has been created, it is never deleted from the directory (except in the case of a username change). NetIDs are also never reassigned. Due to the permanence of NetIDs, NetID authentication alone should never serve as the sole authorization criteria for access to protected applications or resources. Coarse-grained authorization decisions can be made using the activeStudent or activeEmployee attributes (see below) or, preferably, via the use of attributes provisioned by the Enterprise Directory Service.

Any particular NetID entry may be associated with student, employee, or DSV affiliations—or a combination of these (typically student/employee). Affiliates are indicated as "active" (for each affiliation category) if said affiliation is "active" within the appropriate system of record (SIS or PSOS). For example, a person may have been a student last semester, graduated, and become an employee. She would have a single NetID account with both student and employee affiliations, but only the employee affiliation would be designated as "active". DSV "active" status can be ascertained by examining the expirationTimestamp attribute. More "fine-grained" information, for all types of affiliates, may be retrieved from the Enterprise Directory Service.

A "self-service" web application for NetID creation and maintenance is available at http://netid.arizona.edu/. To authenticate to the NetID creation process, students need to know their student ID, assigned PIN, and birth date; employees need to know their employee ID, the last six numbers on their CatCard, and their birth date.

Details regarding inclusion, creation and revocation for students, employees and department sponsored visitors (DSVs) follow.

Students
NetID eligibility is based on data in the university's system of record for students, UAccess Student (PeopleSoft SA). Data that support the tracking of incoming students (admits) and the permanent academic record of students are used to determine eligibility to create NetID credentials.

A student is able to create a NetID if one of the following is true:

- The student has been previously enrolled, but is no longer active
- The student is currently enrolled
- The student is admitted for a future term

A student is considered "active" if:

- The student has an "active" program status in PeopleSoft for at least one career program plan (CPP)
- The student is not in an "admit" state for a future term
**Employees**

Employee inclusion in the NetID directory is based on data in the university's system of record for employees, UAccess Employee (PeopleSoft HCM). Data that support the tracking of employment status (status code and status begin date) are used to determine eligibility to create NetID credentials.

An employee is able to create a NetID if:

- The employee currently has a job record in PeopleSoft with a status code other than “T” (terminated) or “D” (deceased).

An employee is considered “active” if:

- The employee currently has a job record in PeopleSoft with a status code of “A” (active) or “W” (short work break).

**Department Sponsored Visitors (DSVs)**

Inclusion of DSVs is currently an ad hoc process, as DSVs are neither defined nor tracked in either the student or employee systems of record. Additionally, the DSV process does not include those individuals who fall under the guidelines for University Affiliates and Associates—which are defined and tracked in UAccess Employee as “Persons of Interest” (POIs). DSV records are created by assigned “department sponsors”, who are responsible for communicating the necessary information needed for NetID creation to their constituents, as well as for the annual renewal process (DSV NetIDs have a maximum expiration time of 1 year). The Account Administration group within UITS' 24/7 Support Center is responsible for the department sponsor registration process.

**NetID Schema**

The NetID directory schema is based upon a single LDAP objectclass—`uaAuthAccount`—which, in turn, is based upon the standard `account` objectclass. The `uaAuthAccount` objectclass is defined as follows:

```
objectclass uaAuthAccount
  oid 1.3.6.1.4.1.5643.2.1.0
  superior account
  requires userPassword, dbKey
  allows employeeld, studentld, emplld
  activeStudent, activeEmployee, secretHint, secretAnswer,
  userpasswordModifyTimestamp, uidModifyTimestamp,
  expirationTimestamp, affiliateType
```
The attribute definitions are as follows:

**activeEmployee (OID: 1.3.6.1.4.1.5643.2.0.1)**

Boolean flag (1 or 0) indicating whether an employee entry is currently defined as “active” (refer to the “Inclusion and Activity Rules” section). If this attribute is not present, the person is not affiliated as an employee.

**activeStudent (OID: 1.3.6.1.4.1.5643.2.0.0)**

Boolean flag (1 or 0) indicating whether a student entry is currently defined as “active” (refer to the "Inclusion and Activity Rules" section). If this attribute is not present, the person is not affiliated as a student.

**affiliateType (OID: 1.3.6.1.4.1.5643.2.0.12)**

If the NetID belongs to a DSV (see Departmental Sponsored Visitors (DSVs) above), this attribute will contain a code indicating the affiliation of the DSV. This affiliation code may be used, at an application’s discretion, to make rudimentary authorization decisions. The set of codes is as follows:

- AU – UA Alumni
- BV – Vendors and Business Partners
- CA – Conference Attendees
- CI – Independent Contractors, Artists, and Performers
- ED – Dependents of UA Employees
- ES – Spouses of UA Employees
- GA – Government Agency Staff
- II – Inter-Institutional Staff and Faculty
- RP – Religious Center Personnel
- SI – Students Incomplete
- SN – Non-matriculated Students
- TE – Employees of Temporary Agencies
- UF – UA Foundation Members
- VC – Campaign Volunteers
- VS – Sponsored Volunteers

**dbKey (OID: 1.3.6.1.4.1.5643.2.0.2)**

String containing the NetID holder’s unique identification number (UA_ID) within the University Information System. This identifier may be used to retrieve additional information via the Enterprise Directory Service, or other systems keyed on UA_ID.

**emplId (OID: 1.3.6.1.4.1.5643.2.0.4)**

DEPRECATED. String containing the old PSOS employee identification number (EID) of the NetID holder. New employees hired after September 28, 2009 (the date PeopleSoft HCM went live) will not have this attribute. This attribute is currently maintained for cross-reference purposes only, and may be removed in the future.

**employeeld (OID: 1.3.6.1.4.1.5643.2.0.21)**

The converged employee/student primary identifier assigned by the PeopleSoft SA/HCM systems. One unique EMPLOYEEID is assigned to each person, and is synchronized between the two PeopleSoft instances. This attribute replaces the employeeld and studentld attributes.

**expirationTimestamp (OID: 1.3.6.1.4.1.5643.2.0.11)**

String which, if present, indicates the expiration date and time (UTC) of a DSV NetID. This is the date as of which the DSV’s sponsorship is expired.
secretAnswer (OID: 1.3.6.1.4.1.5643.2.0.7)
String containing an MD5 hash of the user-supplied answer to the question contained in the secretHint attribute.

secretHint (OID: 1.3.6.1.4.1.5643.2.0.6)
String containing the user-selected “secret question,” which must be answered by the NetID holder with the contents of the secretAnswer attribute in order to reset a forgotten NetID password.

studentId (OID: 1.3.6.1.4.1.5643.2.0.5)
String containing the SIS/Matrix-supplied student identification number of the NetID holder. This identifier will be deprecated in the near future, as students are migrated to using their 8-digit EMPLIDs in lieu of the 9-digit SID.

uid (OID: 0.9.2342.19200300.100.1.1) [Defined in objectclass account]
String containing the unique userid of the NetID holder. This string is what the NetID holder associates with the “NetID”

uidModifyTimestamp (OID: 1.3.6.1.4.1.5643.2.0.9)
String which, if present, contains the last time (UTC) the NetID holder changed her uid (username).

userpasswordModifyTimestamp (OID: 1.3.6.1.4.1.5643.2.0.8)
String which, if present, contains the last time (UTC) the NetID holder changed her userpassword. This attribute can be utilized by applications that wish to deny access if the user has not changed her password within a certain period of time. The NetID system enforces periodic password expiration, based on the calculated strength of the user’s password.

Using the NetID Service

Web-based Services
Web-based services requiring NetID authentication must use the WebAuth service (see https://netid.arizona.edu/apidocs/webauth/) or Shibboleth (see http://sia.uits.arizona.edu/shibboleth). The following section describes methods for connecting directly to the NetID directory, which should be used only for non-web-based services.

Connecting to the Directory
Connecting to the NetID Directory for authentication should be done only for non-web-based services.

The first step in making use of the NetID directory is to establish a connection to the NetID LDAP server: netid.arizona.edu:636. In order to connect to the directory, certain prerequisites must be met. The first of these is to register the IP address(es) your application will be connecting from with the NetID administrator—the NetID directory will return information only to authorized clients (see “Additional Information” at the end of this document for details on obtaining access). Another prerequisite is that the connection be established over SSL—make sure that the application environment you are using supports LDAP over SSL. Additionally, your application should ensure that the NetID and password are not transmitted in clear text (i.e., unencrypted) at any time—typically, this is accomplished by using the “HTTPS” protocol for web applications, or similar SSL-tunneled protocols for non web-based applications. Under no circumstance should the value of the NetID password be saved by the application.

In order to connect to the directory, an LDAP “authenticated bind” operation must be performed—typically as the user you wish to authenticate. In certain situations, an application may be given a special “distinguished name”, which may be used to bind to the directory in order to view attributes that are normally hidden. Please refer to the “Code Examples” section in order to see how to perform an LDAP/SSL bind within different environments.

Note: Different LDAP/SSL implementations may allow you to specify whether you wish to verify the X509 certificate presented by the server. If you wish to verify the NetID directory’s certificate, please be aware that it is signed by the DigiCert Global Root CA. Your SSL software must be able to verify certificates signed by this CA. Instructions for adding
this certificate under OpenSSL are provided in Appendix A; procedures for doing this within other development environments will vary.

**Searching the Directory**

Once bound (with credentials) to the directory, the NetID holder is considered “authenticated.” This may be all that is necessary for many applications. Some applications may desire to retrieve additional information about the NetID holder (e.g., from UIS); in order to do this, an LDAP search operation must be performed.

All NetID directory searches should be initiated from the following base point:

```
ou=Accounts,ou=NetID,ou=CCIT,o=University of Arizona,c=US
```

Searches may be performed with a scope of either LDAP_SCOPE_ONELEVEL or LDAP_SCOPE_SUBTREE. Please refer to the "Code Examples" section in order to see how to perform an LDAP search operation in different environments.

**Retrieving Entry Attributes**

When bound to the directory as a NetID holder, the following entry attributes (described in the “NetID Schema” section) may be retrieved (all may not be present, depending on affiliation):

- dbKey
- uid
- studentId
- employeeId
- activeStudent
- activeEmployee
- affiliateType
- expirationTimestamp
- userPasswordModifyTimestamp

Please refer to the “Code Examples” section for examples of retrieving NetID directory entry attributes in different environments.

**Access Control Rules**

Entries in the NetID directory are protected against general, unauthenticated, access via access control instructions (ACI’s). Information is not accessible to IP addresses unregistered with the NetID service, and only a NetID holder’s own entry is viewable (i.e., an authenticated NetID user cannot access another NetID’s information). Special "application administrator" accounts, and the directory administrator, have broader access rights.

The following table illustrates the access permissions (by user type) for the various *uaAuthAccount* attributes:
<table>
<thead>
<tr>
<th>secretHint</th>
<th>All</th>
<th>RSC</th>
<th>RSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>secretAnswer</td>
<td>All</td>
<td>RSC</td>
<td>RSC</td>
</tr>
<tr>
<td>affiliateType</td>
<td>All</td>
<td>RSC</td>
<td>RSC</td>
</tr>
<tr>
<td>uidModifyTimestamp</td>
<td>All</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>expirationTimestamp</td>
<td>All</td>
<td>RSC</td>
<td>RSC</td>
</tr>
<tr>
<td>userPasswordModifyTimestamp</td>
<td>All</td>
<td>RSC</td>
<td>RSC</td>
</tr>
</tbody>
</table>

Permissions: R = read, S = search, C = compare, W = write
**Code Examples**
This section will provide some examples of common LDAP operations needed to use the NetID service. Each example will be presented in 3 different environments:

- Perl 5.6.1, using the Net::LDAPS module
- Java (Sun JDK v1.4b2), using JNDI
- Visual Basic Script (VBS), using Microsoft’s ADSI 2.5
Perl / NET::LDAPS

- Step 1: Binding to the directory

  ```perl
  use Net::LDAPS qw(:all);

  # Get user dn/passwd from command-line args
  if ($#ARGV != 1) { die "Usage: ldaptest.pl <username> <password>\n"; }
  $user = $ARGV[0];
  $pw = $ARGV[1];

  # Connect to LDAPS port
  $ldaps = new Net::LDAPS('netid.arizona.edu',
      port => 636,
      verify => 'require',
      capath => '/usr/local/cacerts/') or die "Cannot connect!\n";

  # Bind as NetID user
  $mesg = $ldaps->bind(
      dn => "uid=$user,ou=Accounts,ou=NetID,ou=CCIT,o=University of Arizona,c=US",
      password => $pw);
  $mesg->code && die "Authentication failed: " . $mesg->error . "\n";
  ```

- Step 2: Searching the Directory

  ```perl
  # …bind has already occurred...
  $mesg = $ldaps->search (  # perform a search
      base => "uid=$user,ou=Accounts,ou=NetID,ou=CCIT,o=University of Arizona,c=US",
      filter => "(objectclass=*)",
      scope => "base",
      attrs => ['dbkey', 'activeStudent', 'activeEmployee']
  );

  $mesg->code && die $mesg->error;
  ```

- Step 3: Retrieving Attributes

  ```perl
  # …bind and search have been performed…result is in $mesg
  if ($mesg->count < 1) { # search was unsuccessful
    die "Could not find entry\n";
  }

  # iterate over attributes and print them out
  foreach my $attr ($mesg->entry(0)->attributes) {
    print join("\n ", $attr, $mesg->entry(0)->get_value($attr),"\n";
  }
  ```
Java / JNDI

- Step 1: Binding to the directory

```java
/*
 * Demonstrates how to perform a simple, authenticated bind to an LDAP
 * server over SSL, using JNDI
 */

import javax.naming.*;
import javax.naming.directory.*;
import java.util.Hashtable;

/**
 * usage: java LdapExample <username> <password>
 * 
 **/

class LdapExample {
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            throw new Exception("Usage: LdapExample <username> <password>");
        }
        try {
            // Set up the environment for creating the initial context
            Hashtable env = new Hashtable();
            env.put(Context.INITIAL_CONTEXT_FACTORY,
                    "com.sun.jndi.ldap.LdapCtxFactory");
            env.put(Context.PROVIDER_URL,
                    "ldap://netid.arizona.edu:636/ou=Accounts,ou=NetID,ou=CCIT,o=University%20of%20Arizona,c=US");
            env.put(Context.SECURITY_AUTHENTICATION, "simple");
            env.put(Context.SECURITY_PRINCIPAL,
                    "uid=" + args[0] +
                    ",ou=Accounts,ou=NetID,ou=CCIT,o=University of Arizona,c=US");
            env.put(Context.SECURITY_CREDENTIALS, args[1]);
            env.put(Context.SECURITY_PROTOCOL, "ssl");

            // Create initial context
            DirContext ctx = new InitialDirContext(env);
        }
    }
}
```

- Step 2: Searching the Directory

```java
/*
 * Initial context has been established and bind performed...
 */

// Specify the ids of the attributes to return
String[] attrIDs = {"dbkey", "activeStudent", "activeEmployee"};

// Get the attributes requested for specified entry
Attributes attrs = ctx.getAttributes("uid=" + args[0], attrIDs);
```
Step 3: Retrieving Attributes

/*
 * The attributes for the entry are contained in the Attributes object 'attrs'.
 * Iterate over all attributes and print them out.
 */

if (attrs == null) {
    System.out.println("No attributes");
} else {
    /* Print each attribute */
    for (NamingEnumeration ae = attrs.getAll(); ae.hasMore();)
        Attribute attr = (Attribute)ae.next();
        System.out.println("attribute: " + attr.getID());

    /* print each value */
    for (NamingEnumeration e = attr.getAll(); e.hasMore();
        System.out.println("tvalue: " + e.next()));
}

// Close the JNDI context when we're done
ctx.close();
} catch (Exception ex) {
    ex.printStackTrace();
}
**VBS / ADSI**

- **Step 1: Binding to the directory and retrieving object**

  ```vbs
  On Error Resume Next
  
  If Wscript.Arguments.Count < 2 Then
    Call ShowUsage()
    WScript.Quit(1)
  End If
  
  Set oProvider = GetObject("LDAP:"
  
  Rem Pass ADS_USE_SSL flag to IADs::OpenDSObject Method
  Set oUser = oProvider.OpenDSObject( ___
   "LDAP://netid.arizona.edu:636/c=US/o=University of Arizona/ou=CCIT/ou=NetID/ou=Accounts/uid=", & ___
   WScript.Arguments(0), ___
   "uid=" & WScript.Arguments(0) & ",ou=Accounts,ou=NetID,ou=CCIT,o=University of Arizona,c=US", & ___
   WScript.Arguments(1), ___
   ADS_USE_SSL)
  
  If Err <> 0 Then
    Select Case Err.Number
      Case -2147023570
        WScript.Echo "Invalid Password"
      Case -2147016656
        WScript.Echo "Invalid Username"
      Case Else
        WScript.Echo "Undefined Error: " & Err.Number
    End Select
    WScript.Quit(1)
  End If
  ```

- **Step 2: Retrieving Attributes**

  ```vbs
  Rem ADSI lets us retrieve attributes as properties of the IADs instance
  WScript.Echo "dbKey = " & oUser.DbKey
  Wscript.Echo "activeStudent = " & oUser.ActiveStudent
  Wscript.Echo "activeEmployee = " & oUser.ActiveEmployee
  
  WScript.Quit(0)
  
  Private Sub ShowUsage()
    WScript.Echo "CSCRIPT testldap.vbs <Username> <Password>" & vbCrLf
  End Sub
  ```

  12
Additional Information

- UA NetID WebAuth Service - https://netid.arizona.edu/apidocs/webauth/
- Enterprise Directory Service (EDS) – http://sia.uits.arizona.edu/eds
- Shibboleth – http://sia.uits.arizona.edu/shibboleth
- Net::LDAP Perl module - http://perl-ldap.sourceforge.net/

To request access to WebAuth or the UA NetID directory, you may fill out the request form available at https://bugs.ccit.arizona.edu/netid-request/. For technical assistance, please write to: netid-admin@listserv.arizona.edu

Appendix A
Instructions for installing the NetID Directory’s Certificate under OpenSSL:

If you are using the OpenSSL toolkit in your development (whether in conjunction with the Perl Net::LDAP module, or another environment), you must add the DigiCert High Assurance EV Root CA to your certificate store in order to successfully verify the certificate presented by the NetID directory server during the SSL handshake. Certificate verification ensures that the X509 certificate presented by the server is legitimate. The steps to accomplish this are as follows:

1. Download the DigiCert High Assurance EV Root CA cert from https://www.digicert.com/testroot/DigiCertHighAssuranceEVRootCA.crt. Make sure to save the file to disk, and not install it in your browser.
   a. Note: As of 05/27/2010, the certificate is in PEM (base-64 encoded) format, although the rest of DigiCert’s certificates on their root certificate page are in DER (binary) format. After downloading the certificate, check to see if it is in a binary format—if it is, you’ll need to perform Step 2; otherwise, proceed to Step 3.
2. (See 1a. above) Convert the certificate from DER to PEM format:
   $ openssl x509 -inform DER -outform PEM -in DigiCertHighAssuranceEVRootCA.crt -out DigiCertHighAssuranceEVRootCA.pem
3. Copy the file to the certificates directory within your OpenSSL installation, and protect appropriately. For example:
   $ cp DigiCertHighAssuranceEVRootCA.pem /usr/lib/ssl/certs
   $ chown root:root /usr/lib/ssl/certs/DigiCertGlobalCA.pem; chmod 444 /usr/lib/ssl/certs/DigiCertHighAssuranceEVRootCA.pem
4. Change to the aforementioned certificates directory and perform the following command, to allow OpenSSL to identify the certificate by its subject name:
   $ ln -s DigiCertHighAssuranceEVRootCA.pem `openssl x509 -noout -hash < DigiCertHighAssuranceEVRootCA.pem`.0
Changelog

- 2/14/2002 – GDW : Version 1.0
- 2/20/2002 – GDW : Version 1.01. Added Appendix A; fixed errors with Perl Net::LDAPS example.
- 3/28/2002 – GDW : Version 1.02. Fixed typo (o=UniversityofArizona) in DN passed to OpenDSObject() in the VB/ADSI example code; removed ADS_FAST_BIND flag from ADSI sample code, as it forces anonymous bind
- 1/12/2007 – GDW : Version 1.03. Numerous changes; updated PSOS status code information, added DSV info, changed affiliate code descriptions, updated contact info, updated Thawte cert install instructions.
- 8/24/2007 – GDW : Version 1.05. Updated PSOS Status code information to indicate that employee w/ “B” status are considered active employees.
- 2/28/2010 – GDW : Version 1.06. Made changes for migration to PeopleSoft, updates for new permanent NetID policy