Oregon Department of Human Services
Systems Architecture

“PH Application Access Control”

It is in accordance with Systems Architecture that DHS build integrated, interoperable, secure, and scalable solutions that can withstand change.

Security being a critical application component, this document describes how the above mentioned goals are achieved through architecture that decouples an application's Authentication & Authorization from any one solution.

Systems architecture has defined an Access Control API (Application Program Interface) for integrating Public Health applications.

With a pluggable security model Public Health is able to replace the API with platform specific implementations without requiring significant application development rework.

Application developers are able to code an application once, are not required to make code changes when backend platform changes occur, and have no need for extensive training on backend solutions.

OIS is able to focus on solving the business problem at hand, spend less time in providing a solution, and the risk associated with change is significantly reduced.

The following API is currently in production and applications are continually being integrated.

Walt Davis
Systems Architect
Walt.L.Davis@state.or.us

Features

* When from the same browser, user is authenticated, and user browses to a different application then is not required to re-authenticate.

Role-Based Access - Supports holistic use of roles and insures that users only have accesses to functionality or data in which they have been authroized.

Stores Common User Data - Only stores user data common to all applications and is RFC2251-RFC2256 and RFC1274 compliant.

Shared Data Repository - Applications are integrated to use the same repository for creating, updating, and maintaining common user data.

Reusable API - Defines a standard API (application program interface) whereby developers can easily access the repository to authenticate and authorize user access to an application.

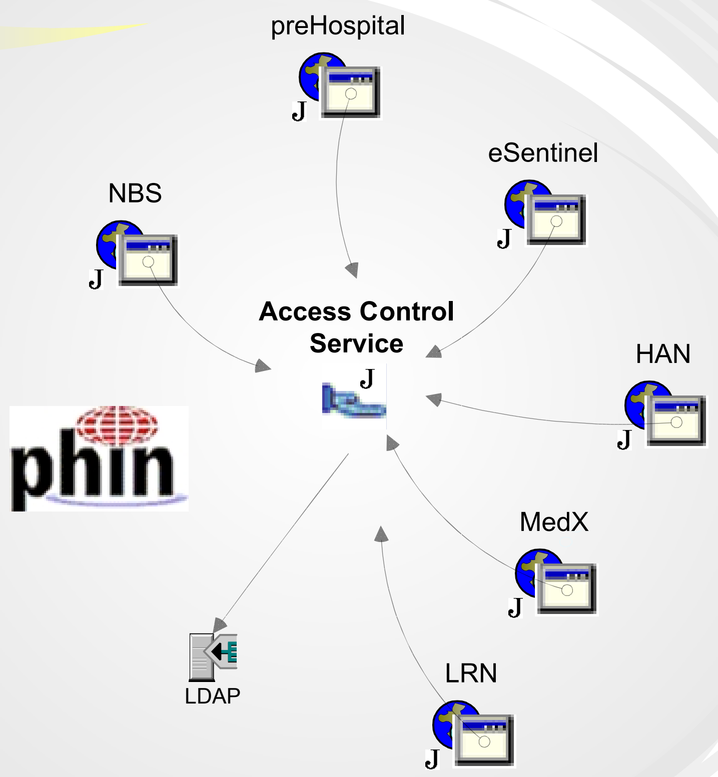
Single-Sign-On* and Single Credentials Set - Once a user has authenticated, he/she can access other applications without having to re-authenticate.

Meets or Exceeds DHS Security Policies - Passwords expire every 60 days, users can not reuse the previous 10 passwords, passwords must be 8 characters in length and contain both a letter and number …

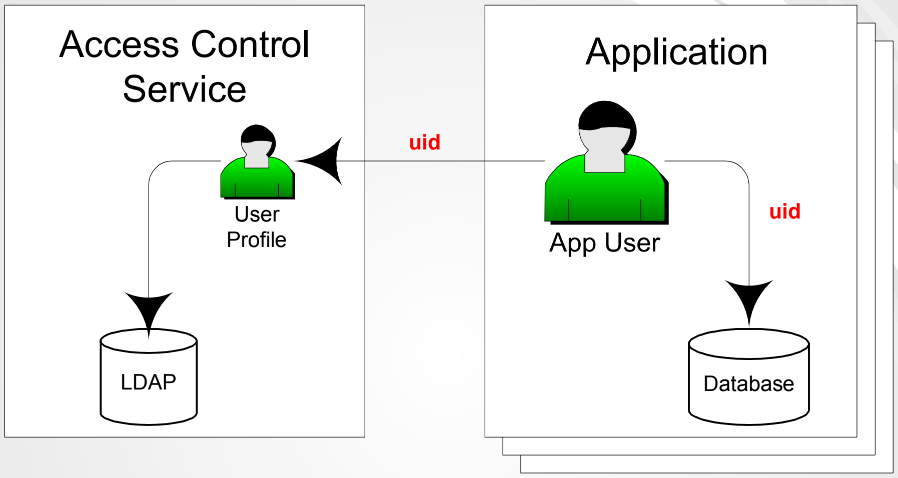
Quasi-Two Factor Authentication - Support for multiple secret question and response as a second authentication factor.

Multi Repository Capable - Has been tested with OpenLDAP, eDirectory, SunONE, and iPlanet; compatible with Tivoli, DB2, Sybase, and other backbends.

Example Implementation

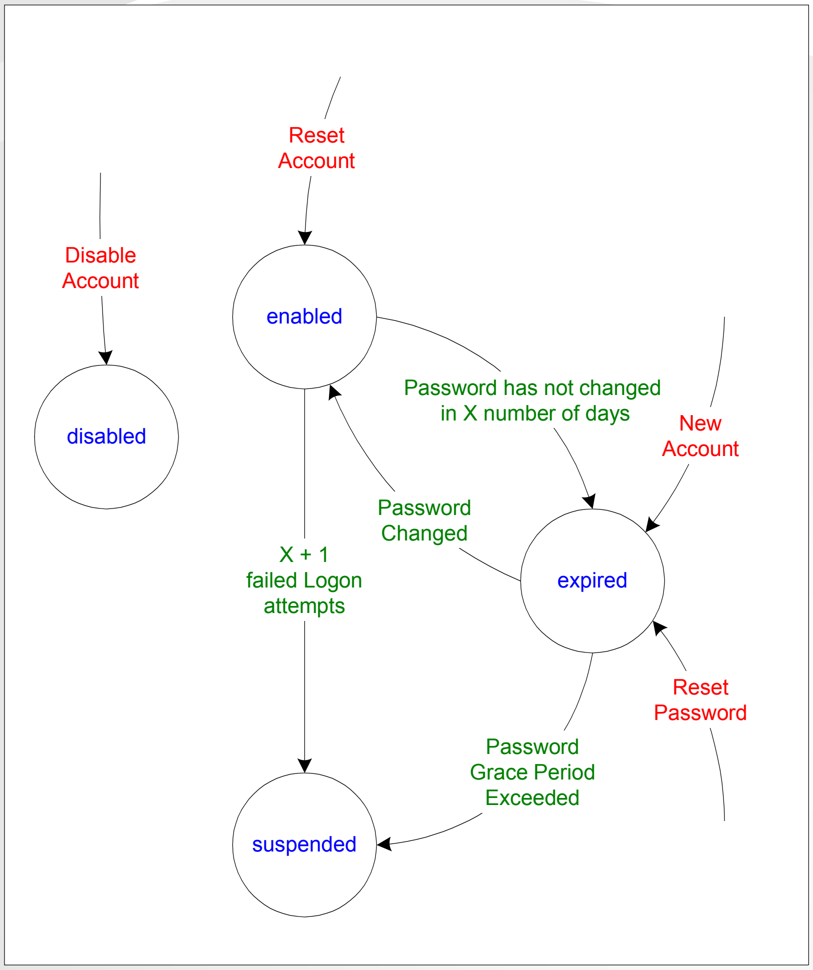


Application Integration



- The Access Control Service stores common data shared by all applications.
- The database stores data unique to a specific application.
- An App User object integrates the application specific data and common data into a single abstraction.
- The App User object is linked to the database object and an User Profile object through a common UID (Unique Identifier).
- Multiple applications are integrated through a common access control service.

Account Status



Reset Account
Disable Account
enabled
disabled
Password has not changed in X number of days
New Account
Password Changed
X + 1 failed Logon attempts
expired
Reset Password
Password Grace Period Exceeded
suspended

State

Disable - user is no longer able to access applications and absolutely no activity on the account is allowed.

Enabled - the account is in good working order and user may freely access applications.

Expired - the users password has expired and it must be change before applications are accessible.

Suspended - occurs when: 1) there has been a number of failed authentication attempts and 2) the user has not changed, within the allowed grace period, their password after initial account creation, account reset, or password reset.

Constants

Days Before Password Expiration - specifies the number of days before user is required to change their password.

Password Grace Period - number of days user has to change their password; applies to newly created accounts, reset accounts, and when the password is reset.

Max Failed Auth Attempts - specifies the max number of failed authentication attempts that once exceeded the account becomes suspended.

Actions

New Account - * creates new user account with password equal to logon ID.

Disable Account - disables the account.

Reset Account - * when an account has been disabled or suspended it must be reset before returning to normal operations.

Reset Password - * changes user password to their logon ID.

** Upon first logon user is required to change password; if user does not do so within the allowed grace period then account is suspended.*

Variables

Number of Failed Auth Attempts - number of consecutive failed auth attempts, is reset to zero upon successful logon.

Date of Last Password Change - is changed when user changes his/her password and upon account reset but not password reset.

Account Status - tracks the state of a users account.

User Profile Demographics

General Info

🔒First Name
🔒Last Name
Middle Initial

🔒eMail

Cell
Fax
Pager

RFC-2256

Work Info

Title
Description

Phone
Street Address
City
County
State
Zip

RFC-2256

Residence Info

Phone
Street Address
City
County
State
Zip

Extension

🔒 *Indicates that a field is required.*

API

Not a complete list of all API signatures!

User Queries

getUser(logonID)
Return user profile for user with the specified logon ID.

getUsersOfApp(appName)
Return those users who have been granted access to the specified application.

getNonusersOfApp(appName)
Return all users who have not been granted access to the specified application.

Authentication

isUserAuthenticated(logonID, sessionIP, sessionID)
Answer true if a user has been authenticated for the specified HTTP session IP address and ID.

authenticateUser(logonID, password, sessionIP, sessionID)
Authenticate that the supplied password matches the users password. If it does and there are no issues with the account then a user profile is returned. Upon success an authentication token is created whereby the user is not required to re-authenticate unless of course the token has timed out.

Authorization

isUserAuthorized(logonID, appName, roleName)
Answer true if a user is authorized to access an application for a the specified role.

grantAccess(logonID, appName, roleName)
Grant the specified user access to an application and role.

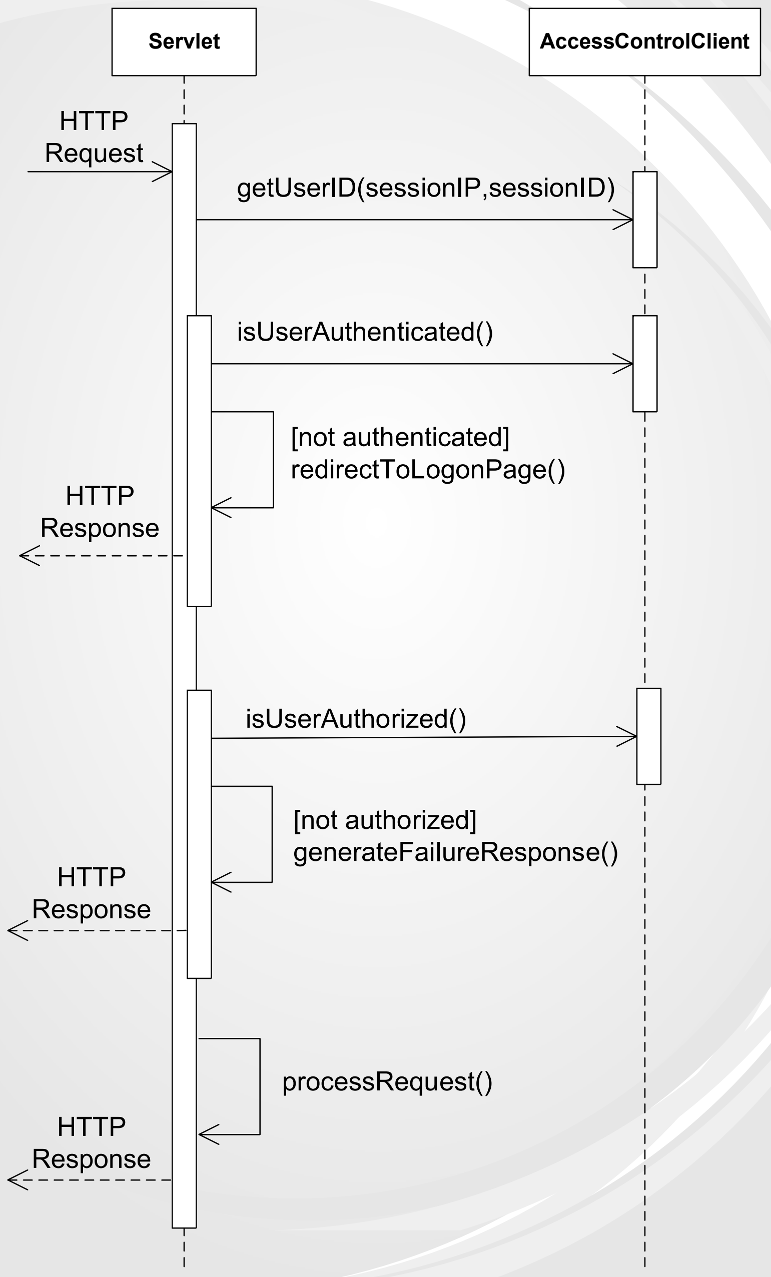
revokeAccess(logonID, appName)
Revoke a users access from the specified application.

Role Queries

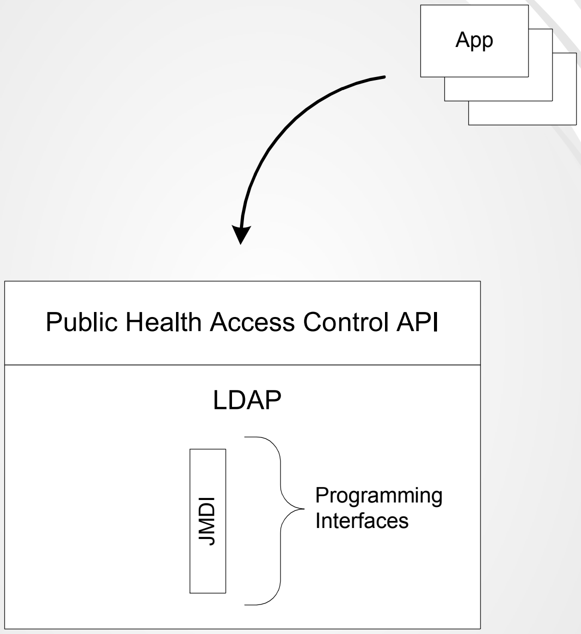
getRolesForApp(appName)
Return all roles for the the specified application.

getRolesForUser(logonID, appName)
Return those roles that a user has been granted access to for the specified application.

Access Control

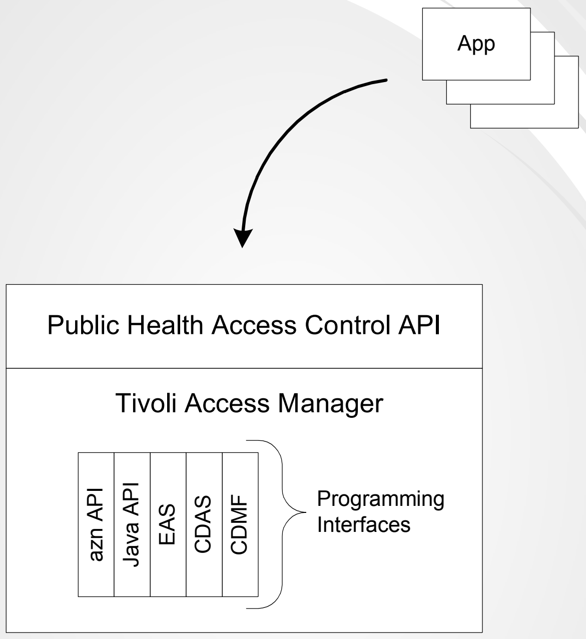


LDAP Integration



JNDI - Java Naming and Directory Interface

Tivoli Integration



aznAPI - ISO 10181-2 & 10181-3 Compliant Interface

Java API - JAAS Compliant Interface

EAS - External Authorization Service

CDAS - Cross Domain Authentication Service

CDMF - Cross Domain Mapping Framework