

# Person Registry September 2007

## Purpose

Accurate identification of individuals who use computer and network systems remains an outstanding issue for many organizations. UC Davis has many existing business and academic systems that perform some level of person identification when new records for individuals are received. Each of these systems is independent of the other campus systems. The Student Information System, for example, has an identification and suspense process for records received for incoming students. The University's Payroll Personnel System has a match process with an interactive process for resolving identity. Other systems that create person records have their own unique methods for resolving the identities of these records.

The campus originally developed a unified name space and account management system in the mid 1990's as its first attempt to resolve identities across disparate systems. Person identity is currently managed in this unified name space. The primary function of this unified name space is to establish a campus logon id and computing account for users. The system extracts records about people entered into other campus systems. The primary sources of information about new persons are the existing administrative legacy systems. Other systems also contribute attributes about persons. The unified name space attempts to create a single record of identity for a person. This system uses a series of match attempts that are dependent upon the source of the records. Reported errors that occur in this system are overmatching and under matching. UC Davis has embarked on a project to create a centralized system in which identity records are consistently evaluated and stored. The project is charged with the implementation of a single authoritative source of identity for individuals. This system will create a unique user identifier (UUID) for each record based upon the evaluation of identity attributes associated with that person. As an example, the attributes of name, social security number, and date of birth could be used to differentiate individual identities. This Person Registry system will replace the existing matching processes that exist in other campus systems.

## Requirements Overview

The Person Registry's sole mission is the creation and maintenance of unique identity records for the population of UC Davis users. It consists of a database, matching routines, APIs (Application Program Interface) for external use, and maintenance programs.

### Database

The registry database contains a small collection of identity attributes.

- **Scalable**  
Must hold hundreds of thousands to millions of records.
- **Extensible**  
Must have the ability to add new matching attributes.

- **UUID Compatibility**  
Unique User Identifiers (UUIDs) should be transformable to the de-facto UUID in use on campus now, the mothraID. The mothraID is simply an eight digit sequence which started at 00000001 when Mothra was first brought on-line in the mid-nineties. It has reached 450,000. The Person Registry UUID is an 12 digit number--an eleven digit number followed by a check digit. A database view or stored procedure should be available to easily transform one into the other.
- **Highly Available**  
Must be available 24/7 with minimal or zero scheduled downtime
- **Responsive**  
Must be capable of handling multiple transactions per second with sub-second response time.

### Matching routines

- **Flexible**
  - "Exact" and "Close" modes. "Exact" only succeeds on an exact match. "Close" would return a UUID if defined criteria were met, for example name and date of birth match, but SSN matches only with a transposition of characters.
  - Can be configured to use a variety of matching criteria
- **High degree of reliability**
- **High performance**

### Application Program Interfaces

- **Secure.** Must meet or exceed University and campus cybersafety standards for sensitive data.
- **Same routines for multiple means of access**
  - SQL
  - Web services
- **Functions**
  - **Match**  
Given Name and ID information returns either:
    - Existing UUID if match is found
    - New UUID if person is new
    - Result code if match is unsure. (Suspense processing is up to client system)

- **Search**  
May return multiple records
- **Update**  
Given UUID and ID information returns:
  - Error code if information conflicts with an existing record
  - Success if record is updated
  - Source systems should be notified of modified UUIDs, for example in case of a merge or split.
- **Delete**
- **Merge** - Multiple entries for what is really a single person are joined.
- **Split** - A single entry that erroneously merges two separate people is split into multiple entries.

## Administrative and Maintenance Programs

- **Web based**
- **Use campus web SSO**
- **Granular authorization.** At the very least, separate query and modify permissions.
- **"Person" Operations**
  - **Query**  
"Exact" and "Close" matches
  - **Insert**
    - Normal - normal insert following existing insert criteria
    - "Override" - overrides existing insert requirements (i.e. insert without a SSN.)
    - "Forced" - insert even if it creates a duplicate.
  - **Merge**
  - **Split**
  - **Update**
  - **Delete**
- **Administrative Operations**
  - **Identifier Validation** - allow admins to add/delete/edit identifiers which can be inserted/stored in the Registry.
  - **Source Validation** - allow admins to define acceptable sources/processes that may interface with the Registry
  - **Result Validation** - allow admins to edit the Result codes returned by the Registry and their descriptions

- **ACL Administration** - administer individual access privileges (Access Control List)

## Operational Requirements

- **Platforms**
  - **Database servers and storage**
  - **Application servers**  
Can use same platforms for web services and administrative access
    - Web server
    - Java container for web services
- **Personnel**
  - **Application developer**
    - Ongoing maintenance and development of person registry itself
    - Integration with other systems
  - **System administration**
  - **Suspense processing**
    - Trusted person to regularly handle records in suspense

## High Level Design

### Overview

The UC Davis Person Registry is an authoritative identity management system for the campus. This system is designed to establish and manage the identification of person records across a variety of campus administrative systems.

### Connecting to the Registry

Source systems can use the Registry for identity management by modifying their code for creating and maintaining person records. These code modifications will interrupt the system's normal record processing by calling the Registry. Connections to the Registry will be made via either database link or web services. All records sent to the Registry must be formatted in XML. A document type definition(DTD) describing the required XML format will be provided to source systems. Results returned to source systems will also be formatted in XML and will follow a similar DTD.

### The Registry Database

At the heart of the Registry is the database that stores all of the person records received from source systems. These records are then used by the Registry for future matching.

addition to storing person records, the Registry stores audit data related to any changes to existing records.

## The Initial Data Load

In order to populate the Registry with current and historical person data from source systems, an initial data load from those systems is needed. This initial load will seed the Registry's database with records from the source system's database and return a unique user id (UUID) to the source system to tie the records together. From the time that the initial load is performed, both databases should be synchronized. Any subsequent modifications to the records in either system will be synchronized via the UUID. Backwards compatibility with MothraID, the existing de-facto campus UUID should be maintained.

## The Registry Matching Routine

All source systems will interact with the Registry via a matching routine that determines whether each person record is a new record or whether it matches an existing record. If this determination cannot be made, the match routine suspends the incoming record for manual review. Each person record that is sent to the Registry is made up of the following identifiers:

- First Name
- Middle Name (optional)
- Last Name
- Social Security Number
- Date of Birth

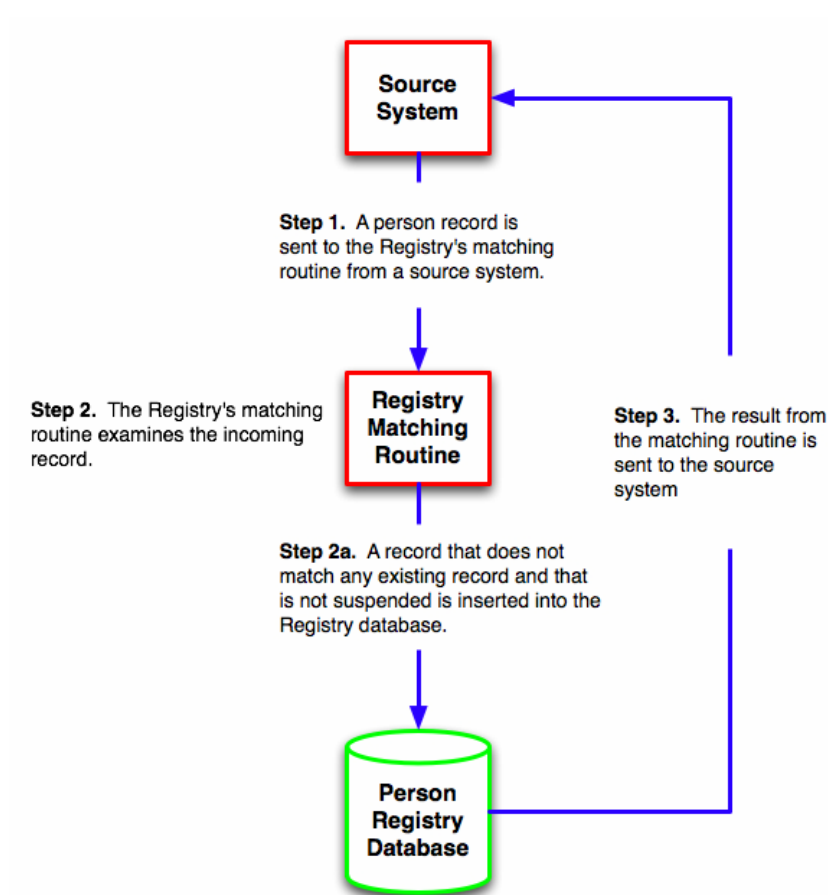
Using these identifiers, the matching routine can uniquely identify a person record and determine if that record already exists in the Registry. Three general outcomes are possible when the matching routine processes these attributes. The record can be a new record, it can match an existing record or it can be suspended.

- **New:** The incoming person record does not exist in the Registry's database. Once this determination is made, the record is immediately inserted into the Registry database and assigned a UUID. This UUID is returned to the source system where it is stored locally to enable future references to that record.
- **Match:** The incoming person record matches an existing record in the Registry. The UUID of the matching record is returned to the source system.
- **Suspend:** The incoming person record cannot be unambiguously determined to be either a matching record or a new record. A suspend code explaining the circumstances of the suspend is returned to the source system.

## Registry Process Diagram

The process by which person records are sent to the Registry and a result sent back to the source system is a three-step process. An additional step occurs only for records that are new to the Registry.

- **Step 1.** The source system creates a person record and sends that record to the Registry.
- **Step 2.** The Registry's matching routine receives the record and attempts to match it to an existing Registry record. If a matching record is found, that UUID is returned. If a record cannot be unambiguously classified, then a suspend code is returned. If no matching record is found, then Step 2a will occur. Otherwise, Step 2a is skipped.
- **Step 2a.** If no match is found for the incoming record, that record is immediately inserted into the Registry's database and a new UUID is generated and returned to the source system.
- **Step 3.** Regardless of the outcome of the matching, the result is returned to the source system.



## Updating Registry Records

Registry records can be updated by source systems by using the UUIDs assigned to those records. Since a record could be modified in a way that would create a duplicate, the Registry will ensure that the modified record will not conflict with any existing records before it implements the update. Once the update succeeds, the Registry will automatically notify source systems of the change.

## Registry Security

All interactions with the Registry must be highly secure due to the sensitivity of the data involved in the transactions. Appropriate security measures will be used to protect sensitive data, including:

- All network traffic to and from the Registry will be encrypted.
- The Registry will be behind a strict set of firewall rules.
- The strongest practical authentication methods will be used for database, web services, and administrative access.

## Person Registry and PPS

The goal of the Person Registry is that all key business and academic systems that create or update person records interact with the Registry before writing person information to their business or academic databases. The Registry is responsible for establishing the unique user id (UUID) for the person and providing this unique id to other systems for easy person matching in the future. Resolution of ambiguity of person records is done by the source system offices. Any change to key person attributes occur in both the Registry and the source system. This update of attributes is accomplished via the same Registry API but with the UUID supplied from the source system.

Since PPS is a vital administrative system responsible for creating and updating person information, it is crucial that it be modified to interact with the Person Registry. PPS modifications need to be made to the payroll interface that creates identity within PPS (Independent ID or IID module) and other attribute update sections of the system which will allow it to interact with the person registry. The detailed steps for transactions would then be as follows (see diagram above) :

- **Step 1.** When a new employee arrives to be processed in the PPS system, the employee's match attributes are entered into the PPS IID module which passes them to the Person Registry application programming interface (API).
- **Step 2.** The Registry API receives the attributes and matches the incoming record to an existing Registry record or creates a new person record in the Registry.
- **Step 3. (Match or New Person)** If the result of the match is to create a new person or match an existing person, the person registry application reports the condition of the identity back to the PPS application. The report includes a

generated or existing unique user ID (UUID).

If the registry reported a new person creation, the PPS system would then create the new person locally, generate an employee ID, and assign the entire record to the UUID received from the registry API.

If the registry reported a match to an existing person, the PPS system would apply the new person to the existing PPS record via the UUID already contained on the PPS system.

- **Step 3. (Unclear Match)** If the result of the match is unclear, the Person Registry returns a code specifying how the record was ambiguous. Since PPS only processes people real-time, the Person Registry will return existing records that may be potential match candidates. The PPS user would then evaluate the returned registry records to determine the existence of a match or to create the record as a new person.

## Person Registry and Banner

Registry and Banner - waiting to talk to Banner folks.

## Development and Implementation

The difficult issue with the Registry is the technique by which it will integrate with source systems for implementation. Possible implementation strategies discussed were:

**Option # 1.** Update the Registry at a source system's person record create time. That is, change all programs that exist in business source systems that create and update person identity attributes. Change the programming logic so that the source system will hand off to the Registry to search and create in the Registry first. The source system would receive back from the Registry either a UUID (new person) or an error code (duplicate, etc.).

**Pro:** Strong real-time integration between the Registry and source systems.

**Con:** Intrusive for source systems.

**Option # 2.** Source systems update the Registry subsequent to person record create time. This would entail a change to all programs that create and update identity attributes. After these programs have completed locally, they would need to push the new and updated records to the Registry. Returned UUIDs and error messages can then be linked to local person identities and processed.

**Pro:** Less intrusive into a source system's programming.

**Con:** Less than real time integration between the Registry and a source system.

**Option # 3.** The Registry initiates a pull (or extract) of records from source systems on a TBD frequency to create new records and update existing records based on source system entries.

**Pro:** Unobtrusive into the programming of a source system.

**Con:** Not current with the population of campus source systems.

While acknowledging the superiority of option 1, the Registry evaluation team anticipates that option 3 will be the strategy most favored by source systems since it is currently the method Mothra uses. However, the team is prepared to support all three options by providing implementation consultation and on-going enhancements to the Registry for source systems. Eventually it may be possible to transition source systems to option 1.

In light of the least intrusive option (# 3), the team did not decide on a single method by which the Registry will detect new identity records in a source system. However, the most obvious method, and possibly the best, is to integrate the Registry with Mothra.; Since Mothra is already performing a Registry-like role, it would be easy to break that role out of Mothra and have it communicate with the new Registry. The existing source system Mothra feeds could continue as normal at first.

Current source system Mothra feeds include person information such as name, ssn, and birthdate. For the new system to work they would need to add a UUID to this feed. When Mothra retrieves these records, if it detects a UUID, then it already knows exactly who this person is. If the UUID does not exist in the feed, then Mothra knows it needs to contact the Registry to generate a new person record and thus a UUID. The newly generated UUID would be returned to the source system, which would keep track of it locally and feed it to Mothra during all subsequent feeds. This process can be gradually implemented:

1. Install the Registry inside Mothra. This may be configured as the same server but a different database instance.
2. Modify Mothra's processing of batch updates to invoke the Registry match process thus creating UUIDs for each entry. As the Registry creates each identity record, it will return the UUID back to Mothra but not push the UUID back to source systems.
3. Source systems will be instructed to modify their data-stores on their own time with the expectation that they will store the UUID when they are ready.
4. Source systems populate their UUID fields created in step #3. This can occur either by running a special batch processes to pull data from the Registry or we can modify the Mothra batch updates to return a UUID back to the source systems.
5. For Mothra's regular batch updates from source systems, Mothra will pull each UUID with other data it normally retrieves. If Mothra retrieves a record without a UUID, Mothra can query the Registry for a potential match based on other identity attributes. If the match comes back with a UUID, Mothra can store the UUID and accompanying data.

**Attribute Ownership.** The team did not decide if the Registry should accept the responsibility to detect updates to identity attributes that occur in source systems. One option is for source systems to always keep a record that describes old and new values for a changed person attribute, identifiable by the UCD UUID already obtained.

Alternatively the source system could be authorized to push the changes directly to the Registry's database by invoking the Registry's API that employs an update routine.

**Suspended Records.** Outstanding issues such as suspended records - records for which the Registry cannot make a definitive match and must return to the source system with an error message - provide several solutions as well. Currently, data stewards for the source systems are responsible for investigating and resolving any partial matches that become suspended records. Data stewards or their delegates are informed via error messages that suspected records have partially matched with other existing records and therefore are in error.

The Registry Evaluation team, while not charged specifically with the business process for handling suspended records, is in agreement with the current business process. Also the team is impressed that the Registry's match routine is stringent and conservative but provides overrides for flexibility. However, the team recognized that the current process of resolving suspended records might require direct access to the Registry for Data Stewards in the future. The original Person Registry team anticipated this likelihood.

**The Crux of the Matter.** The difficulty with the Registry has always been one of integration with existing source systems, some of which are purchased products with on-going responsibilities for vendor updates that may threaten local modifications. Typically, these source systems are heavily used systems and the window of time available for them to incorporate API calls to the Registry for the exchange of identity data is limited. This observation is true increasingly for all middleware products, including Enterprise Authorization, Directory services, Distributed Authentication, and applications that are accessed via MyUCDavis.

### **Conclusion and Recommendation.**

1. Employ concrete methods for source systems to integrate with the Registry and outline Middleware consultation/support services.
2. Develop procedures/business rules for changing identity attributes.
3. Re-tool the Registry to work closely/within the Mothra database
4. Install the Registry on a production platform and run parallel Mothra Registry services for a time. Compare results.
5. Replace Mothra Registry functionality with new Registry
6. Promote source systems to gradually phase-in integration of UUID and option #3 and possibly transition towards option #1