OpenRegistry Response To Registry Questionnaire (Rutgers Responses)

IAM Registry questions to evaluate features and functionality against standard business requirements.

Category	Description or Question for solution provider	Response	Link(s) to Documentation

General architecture	Describe how ID match capability is provided by the registry solution. For example, is it (a) an integral part of the solution as provided or (b) must it be integrated with an external ID match engine or (c) can it be provided in some other way?	ID Match is part of the Reconciliation process. There is a generic implementation "reconciler" provided in the Open Source community, which can be extended to meet the institution's specific need. This reconciliation module is a plug-in within the product. It can be integrated with an external ID match with some work effort
		
		In Rutgers, we implemented/extended the generic "reconciler" interface to consider the following Identifiers for matching
		(SoriD, Registry unique ID, SSN, NetID) High Assurance
		(name, DOB) Medium Assurance
		(email, Address, Phone) Low assurance
		
		SoRID: This identifier is created by the authoritative source of record, for example employee ID in HR system or Student ID in Student system
		Registry unique ID (inherited from the legacy system): This is unique IDM identifier that can be used to identify multiple role user (staff/faculty will have the same Registry unique identifier, but different SoRID's)
		SSN: Can be real, pseudo
		NetID: uid generated by the registry and fed back to the System or Records
		
		Conflict means a human intervention is needed
		No match: New record
		Match: existing user
		
		When the SoRID exist and match what is on the registry we route the request to "update person" service layer, otherwise we start the matching process (reconcile) so the order for the high assurance level identifiers are registry unique ID=>SSN=>NetID
		
		In general, we start matching the high assurance identifier, if a match is found, we move across the other high assurance identifiers first and then move down the list to the medium assurance identifiers. If any of the high assurance identifiers has a match, but no match is found in the lower level (medium), a conflict is raised. For example, SSN is matched in the registry, but neither name and DoB is at match, this will return a conflict "SSN matched. Conflicting Data Found: Name did not match" or "SSN matched. Conflicting Data Found: DOB did not match".
		
		If any of the higher assurance identifiers does not match what is on the registry, we move across the other high assurance identifiers and then down to the medium level identifiers. If we find a match for both of the medium level identifiers, we raise a conflict. For example, no match on SoRID, no match on Registry unique ID, no match on SSN, but match on both name and DoB will result in conflict "Found person with same name and DOB but SSN didn't match". So if any of the medium level identifiers did not match, this is consider a new record (no match), for example SSN does not match what is on the registry, Name match, DOB does not match, this is considered a new record
		
		NetID is checked whenever there is no SSN matching or real SSN is provided. If NetID matched what is on the registry, we check medium level identifiers (name and DoB) and if any mismatch, we raise conflict, otherwise if both match, then it is a match. If NetID does not match, we raise a conflict. Example, a former employee came back to take short term class, which does not require to collect SSN from the user. The former employee still remember the NetID. For this user to be reconciled (to match registry record), NetID, Name and DoB has to match
		
		Low level assurance identifiers are only used when SSN is not provided in the data feed (To support SoR that might not require SSN, for example short-term classes). In this case, we check if NetID exist and match what is on the registry as previously stated above. If no new NetID was provided in the data, Name, a high medium level identifiers have to match first and then we do match on the lower level identifiers all have to match, otherwise, it is a new record. Note: This function is still under development
		
		Under Construction: analyzeNonUniqueMatches
		
		protected final int HIGH_CONFIDENCE = 90;
		protected final int MEDIUM_CONFIDENCE = 50;
		protected final int LOW_CONFIDENCE = 25;
			Details are in the code based on Rutgers legacy.
			
			Code can be shared per request

	Describe how groups management (for use with authZ controls and other purposes) is provided. For example, is it (a) handled internally by the solution or (b) integrated with an external group management engine such as Grouper or (c) provided in some other way?	It is the vision to extend the OpenRegistry to data beyond persons. Group Registry is in the Road map for both jasig and Rutgers. OpenRegistry, which is a person registry can be integrated with any external group management engine such as grouper via the provisioning engine. The OpenRegistry, however implemented authorization model utilizing simple group structure to manage the Web Interface to the registry.	https://wiki.jasig.org/display/OR/Group+Use+Cases
Data model	Describe how the registry solution supports an extensible set of attributes about (a) persons, (b) applications or other external resources, and (c) other, arbitrary entities?	The Data model of the OpenRegistry is flexible to support additional attributes/identifiers. The data model is extensible to define new identifier and new data types without the need to create new tables. OpenRegistry supports non-entity person. This is dedicated to handle Service accounts, shared accounts, reserved system accounts, etc... The data model reflects the separation between the person and non-person entities. The service layer, however, deals with it as one virtual entity when assigning identifiers (to avoid identifiers collision). This design took under consideration managing Identity life cycle (provisioning, usage, de-provisioning) and all potential access management related implications for real users (password policy, audit, rules,etc...) while offering IDM tools (governance, work flow, audit) to streamline service account creating As far as the external resource (downstream systems), the data model does not reflect any mapping between the user's role and the external resources (LDAP, Unix, Email service, etc) as of yet. The vision is, there will be a provisioning engine (external to person registry) that will maintain this mapping.	https://wiki.jasig.org/display/OR/Data+Model
AuthZ support	Describe how the registry data model supports defining arbitrary user roles in support of authZ functions.	Roles are derived from the Source of Records and are created in the registry. Role's information are available to the downstream systems for authZ function via OpenRegistry DB views, LDAP, or Message queue	https://wiki.jasig.org/display/OR/Architecture https://wiki.jasig.org/display/OR/Data+Flow

Features	Describe how the registry solution supports audit logging of sensitive transactions, including support for the recording of historical changes made to sensitive data. Describe how this log includes the requester and authorizer identities, and transaction timestamps.	In additional to the application logs, each of the Registry table has a corresponding audit table with timestamps that can be further utilized by any external reporting/auditing engines. Our auditing mechanism has a concept of revisions. Basically, one transaction is one revision (unless the transaction didn't modify any audited entities). As the revisions are global, having a revision number, you can query for various entities at that revision, retrieving a (partial) view of the database at that revision We also added a master file that also captures the requester information with timestamps. The master table for revision has a field 'username'. Auditing Technologies: Inspektr, log4j, Envers from JBoss.	
	Describe how the registry solution supports the secure storage of security questions and answers for use in password recovery.	The Person Registry does not hold any password meta data for now. This is still an open Design question whether we want to include this in the person registry or to have a separate credential registry. The registry, however, stores activation key data (Activation key, End date, start date, lock, expiration date). Activation keys are used to activate the NetID, and to reset Forgotten password We are building an external application "NetID management tool" that will utilize the activation key (via REST calls to the OpenRegistry). The NetID application will also use the security questions and answers, which are not stored in the registry.	https://wiki.jasig.org/display/OR/Activation+Keys https://wiki.jasig.org/display/OR/Generate+Activation+Key https://wiki.jasig.org/display/OR/Verify+Activation+Key
	Is there support for multiple name and address types as well as history? If yes, please describe.	Yes Name types supported: Official name, Preferred name, Formal, FKA, Maiden, Legal Address type: Campus, Home, Office History: Audit tables contains historical data	https://wiki.jasig.org/display/OR/Data+Model
Identity Assurance	Are registration events captured as they occur? Do these events automatically trigger assignment/deassignment of an IAP	IAP data is not being collected by the registry for now	
	Is there support for real time provisioning of Identities/services	Yes, this is done via 1) Camel routing engine - current 2) Active messaging queue - planning	https://wiki.jasig.org/display/OR/Architecture https://wiki.jasig.org/display/OR/Data+Flow https://wiki.jasig.org/display/OR/Open+registry+real-time+integration+with+consumers+implementation+patterns
	Describe how data is processed (batch, web services)	Upstream: Batch: HR system Batch: Student system Web: Guests REST: Update email address, NetId Change, Assign NetID's to non-person entities	https://wiki.jasig.org/display/OR/Data+Flow https://wiki.jasig.org/display/ORUM/RESTful+API
	Is registry dependent on other open source or vendor products? If yes, please provide details.	OpenRegistry is an incubated project hosted by Jasig. It is bound by the Apache License. It is an OpenSource product project built on: SpringFramework 3.0.5 Hibernate.version 3.5.5 Camel version 2.5.0 ActiveMQ version 5.4.2 Tomcat version 6.0.32 Java SDK 1.6 Database: Oracle 11g (this can be any relational data base). Rutgers uses Oracle Please see https://wiki.jasig.org/display/ORUM/License	https://wiki.jasig.org/display/ORUM/License
	Where is the business logic stored? Is there support for delegation to maintain these rules?	The business logic is stored within the service layer code and XML files We are exploring Drools as a rule engine	https://wiki.jasig.org/display/OR/High+Level+Architecture
	How does the registry notify external entities of data changes? (for example name is changed)	1) Updating Response tables via Camel routing - current 2) via AMQ. Services will subscribe to the provisioning engine queues - in planning	
	Is code located in public repository	The OpenRegistry code is hosted by Jasig and is available to public. In Rutgers we overlay the Open Source code and store a Rutgers specific implementation locally and it is not accessible to the public.	https://source.jasig.org/
	How are changes, marketing, etc communicated to public? (wiki, lists, web presence)	This is communicated via: Jasig wiki pages (see links) Openregistry-dev list Openregistry-user list OpenRegistry Jasig Monthly call	https://wiki.jasig.org/display/ORUM/Home https://wiki.jasig.org/display/OR/Home https://wiki.jasig.org/display/JSG/openregistry-dev https://wiki.jasig.org/display/OR/Releases
	Is there proper OSS license?	Yes. see https://wiki.jasig.org/display/ORUM/License	https://wiki.jasig.org/display/ORUM/License
	Is there a clear project lead?	Yes. From Rutgers this is sponsored by the Office of Information Technology	
	Is there an existing project steering committee /governance?	Yes. The OpenRegistry Advisory Committee is a voluntary group asked to review and comment on the progress of the OpenRegistry initiative, in order to provide feedback to active OpenRegistry Developer Institutions.	https://wiki.jasig.org/display/OR/OpenRegistry+Advisory+Committee