RUTGERS

OSIDM4HE

Identity Management

OpenRegistry

Rutgers Implementation

May 04, 2012

Agenda

- Part 1 - Overview
  - Project Overview
  - What Is OpenRegistry
  - Benefits
  - Approach
  - Project Status (Rutgers)
    - Certification 1
    - Certification 2
    - 1st Production
- Part 2 – Component/Service Architecture
  - Component Architecture
  - Data Model
  - Auditing
  - Road Map
- Part 3 – Demo
- Part 4 – Code Review

PART 1

OpenRegistry Overview

Project Overview

- Open source software application
  - Initiated by Rutgers University - Mar 2008
  - Became a Jasig Incubator project – Jan 2009
  - SFU joined the project - Late of 2009

What Is OpenRegistry

- An Open Source identity Management System, a place for data about people affiliated with your institution
- Identity store, but generally NOT authoritative
  - Systems of Record (SoRs) are authoritative for most data (e.g. PeopleSoft, Student System, etc.)
- Identifier assignment (NetID, RCPID, External ID,etc.)
- Identity reconciliation for multiple SoRs. It combines distributed identity information into single identity records

OpenRegistry High Level Benefits

- Capture information about all University populations (e.g. students, staff, faculty, guest, visitors, alumni, retirees, emeritus, continuing ed, etc…)
- Faster propagation of data from SoRs , real time where possible
- Role Base access control
- Automating User Lifecycle Management (On-Boarding, Off-Boarding, Approvals, etc.)
- Regulatory compliance
  - Provide timely audit and attestation
  - Provides long term, sustainable model

OpenRegistry Open Source Approach

- Generic architecture and data model
- Agile development process
- Multiple levels of engagement with the Higher Ed community
  - Discuss: Review design documents, identify gaps and changes
  - Develop: Help write code, documentation, etc
  - Deploy: Run OR as an IDMS (when released)
  - Donate: Contribute resources to help with development and outreach
  - Encourage others to join the work force
- Build on lessons learned from other Jasig successful Open source projects (CAS, uPortal)

Project Status @RU(OpenRegistry Core)

- Generic data model designed and stable
- Core domain objects and base service layer code completed
- Authorization module completed and tested
- Web UI authentication is integrated with CAS
- Reconciliation/Calculation Enhancement completed
- Normalization and Standardization completed
- RESTful API's majority designed and completed
- Identifier assignments majority completed for first Production release

Project Status (Certification1 Release)

- Scope :
  - Initial load from legacy People Data Base (PDB)
  - Student Record system batch feed
  - HR Record (PeopleSoft) system batch feed
  - PDB-Like views (Used for Validation)

Project Status (Certification1 Release)

- Time Line : May, 2011
- **Status: Completed**
- Deployed In Pre Production Environment
- Worked with Real production data from two SoRs (HR and Students)
- Worked In parallel with the legacy People Data Base (PDB)
- Ran Validation Frequently
- Compared results from OpenRegistry repository and legacy PDB
- Used UI to view person for validation
- Used Views for validation

Project Status (Certification2 Release)

- Scope:
  - Help Desk UI's
    - Add/Update person
    - Add/Update role
    - View complete person
    - Generate Activation Key
    - Authorization Model
  - Disclosure
    - Person level
    - Attribute level
  - REST API's
    - Email Address Update
    - NetID change
    - Generate Activation Key

Project Status (Certification2 Release)

- Time Line : September , 2011
- **Status: <u>Deployed</u>.** New Requirements were added
  - Handle Early Faculty/Staff Emergency Role Requests and Approvals (when new Fac/Staff not processed in HR yet)
- Deployed In Pre Production Environment

Project Status (0.9 – Go-Live Release)

- Scope:
  - User management Tool UI
    - Every Thing from Certification 2
    - Split/Join person
    - Authorization Tools (Administration/Delegation)
    - Guest Management
    - Herd Assignments
    - Early Faculty/Staff
    - NetID Change Authorization
  - Registry Reconciliation Enhancements
    - Every Thing from Certification 1 and 2
    - LDAP Real Time Provisioning via Camel Routing
    - Process Students' personal email from SRDB Data Feed
    - New Add Work Flow (Support Early Faculty/Staff Emergency Provisioning requests, not yet in HR system)

Project Status (0.9 – Go-Live Release)

- Scope:
  - RSET API
    - Every Thing from Certification 2
    - Support Of non-Person Entities (Service Accounts)
  - Email Services OpenRegistry Integration
    - RATS Updates Display Email Address via REST API
  - Personal Info/OpenRegistry
    - Update User's Information on Behalf of SoRs (email address, emergency contacts, etc..)
  - Legacy-Like Views
    - Used for Validation /Comparison Only

Project Status (0.9 – Go-Live Release)

- Scope:
  - Legacy BackSync from OpenRegistry
    - Data Feed from PDB to downstream systems will Continue for This Phase
    - Clients integrated with PDB do NOT need to change code for initial release, allowing more transition time
  - Support new ID Card # (RCN)

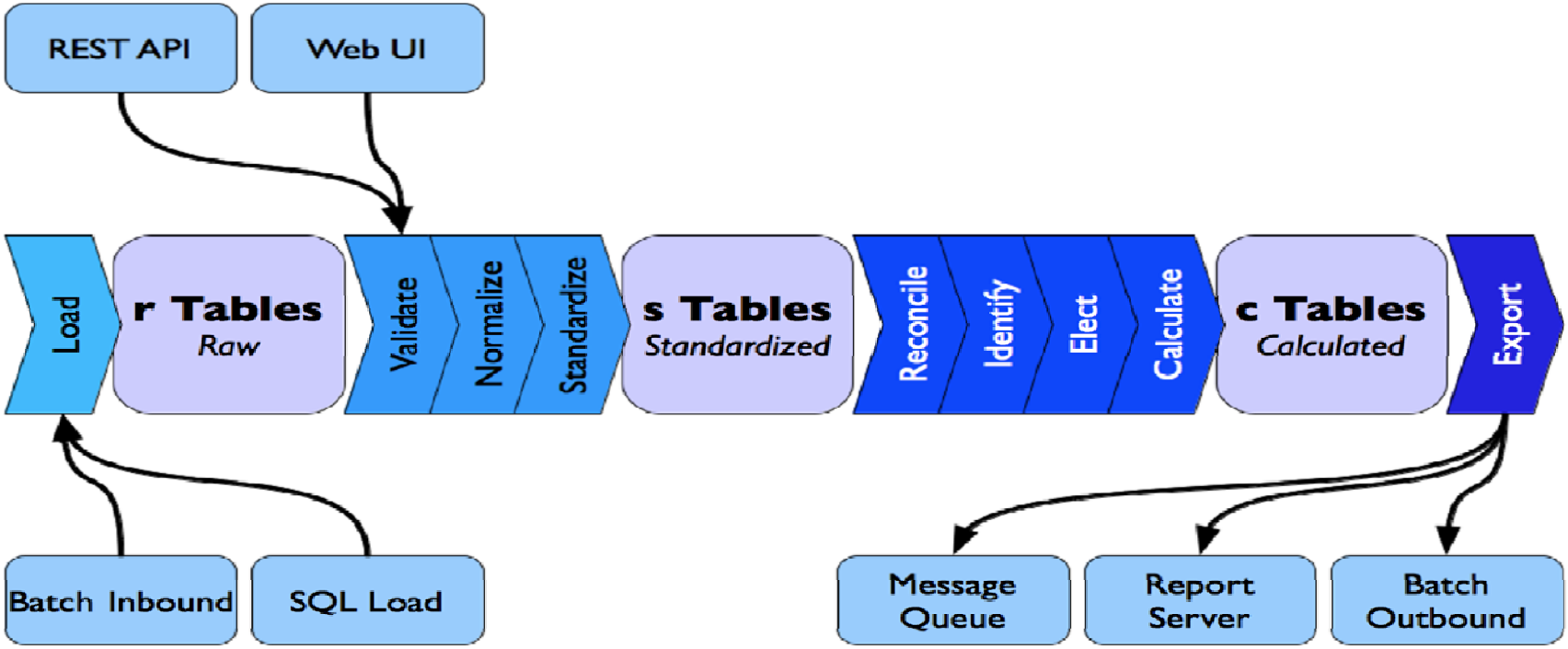
Project Status (0.9 – Go-Live Release)

- Time Line : August , 2012
- **Status: <u>Completed</u>**
  - **System Integration Testing:** May 2012
  - **Target Go-live date :** August 3, 2012

PART 2

Component Architecture

Component Architecture



Component Architecture (Rutgers)

- **Load:** Data is loaded from batch oriented sources into the "raw" (r) tables **rTables**. Data is loaded as is, and represents what was provided by the System of Record (SOR). Data provided from real-time sources does not need to be staged, and is passed directly to Validate

- **SQL Loader** (Raw data) : Processing data in bulk from the Source of Records. For example loading data from HR DW into pre-processing scratch tables

- **rTables:** *Tables used to store raw SoR data provided for processing*

Component Architecture (Rutgers)

- **Validate:** Validation is a check of the inbound data against the defined syntax agreed upon (contract) with the SOR, as well as enforcement of the security model that determines what data the SOR is allowed to assert. For example, GivenName is a required field

- **Normalize:** Normalization is the process of transforming the data according to institution specific rules designed to make all data look the same regardless of how it is represented by the SOR. For example Transforming inbound data from UPPER CASE or other format to Mixed Case

- **Standardize:** Standardization is the process of mapping the data from SOR specific interface definitions to the standard OpenRegistry data model. After standardization is complete, data is stored in the "standardized" (s) tables **s Tables.** For example, Mapping SORID which is “char” type to OpenRegistry as “varchar2”

Component Architecture (Rutgers)

- **sTables: Tables** *used for standardized data, which will be processed for uniqueness and changes*

- **Reconcile:** Reconciliation is the process of matching records from one SOR with those already known to the system. It is key to maintaining a single identity for a person regardless of how many sources they come from. Reconciliation is configurable based on the attributes available in the "standardized" (s) tables, and on a per-SOR basis. That is, SORs that may not provide sufficient quality data for reconciliation to take place may be reconciled differently from those that do

  - 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

Component Architecture (Rutgers)

- **Reconcile (Identity Maching approach):**
  - In Rutgers, we implemented/extended the generic "reconciler" interface, based on the legacy people data base approach (pretty complex). This approach considers the following Identifiers for matching :
    High Assurance: (SorID, Institute unique ID, SSN, NetID)
    Medium Assurance: (Name, DOB)
    Low assurance: (Email, Address, Phone)

  - Identifiers (used for ID matching)
    - SorID : This identifier is created by the authoritative source of record, for example employee ID in HR system or Student ID in Student system
    - Institute unique ID: (Inherited from the legacy system): This is unique IDM identifier that is generated by the registry and fed back to the System or Records . It can be used to identify multiple role user (staff/faculty will have the same Rutgers unique identifier, but different SoRID's)
    - SSN: Can be real or pseudo
    - NetID: User Name, generated by the registry and fed back to the System or Records

Component Architecture (Rutgers)

- **Reconcile (Identity Maching approach):**
  - Match Results:
    - Conflict: a human intervention is needed to resolve
    - No match: New record
    - Match: existing user , update person
  - New Approach
    - **Under Construction**: analyzeNonUniqueMatches
      protected final int HIGH_CONFIDENCE = 90;
      protected final int MEDIUM_CONFIDENCE = 50;
      protected final int LOW_CONFIDENCE = 25;
    - **Looking for new Ideas**

For details OR-Rutgers response for ID match:
https://spaces.internet2.edu/x/p56VAQ

Component Architecture (Rutgers)

- **Identify:** Identification is the process of assigning identifiers to newly added persons. Identifier assignment is configurable based on locally defined formats . Example of identifiers:
  - SSN (pseudo)
  - RUID
  - RCPID
  - IID
  - NETID
  - EXTERNALID
  - EMPLID
  - RCN

- **Elect:** Election is the process of deciding which conflicting data provided by multiple SORs is the "true" data for a person. For example, if two SORs assert a different official name for the same person, only one name can be treated as official by OpenRegistry

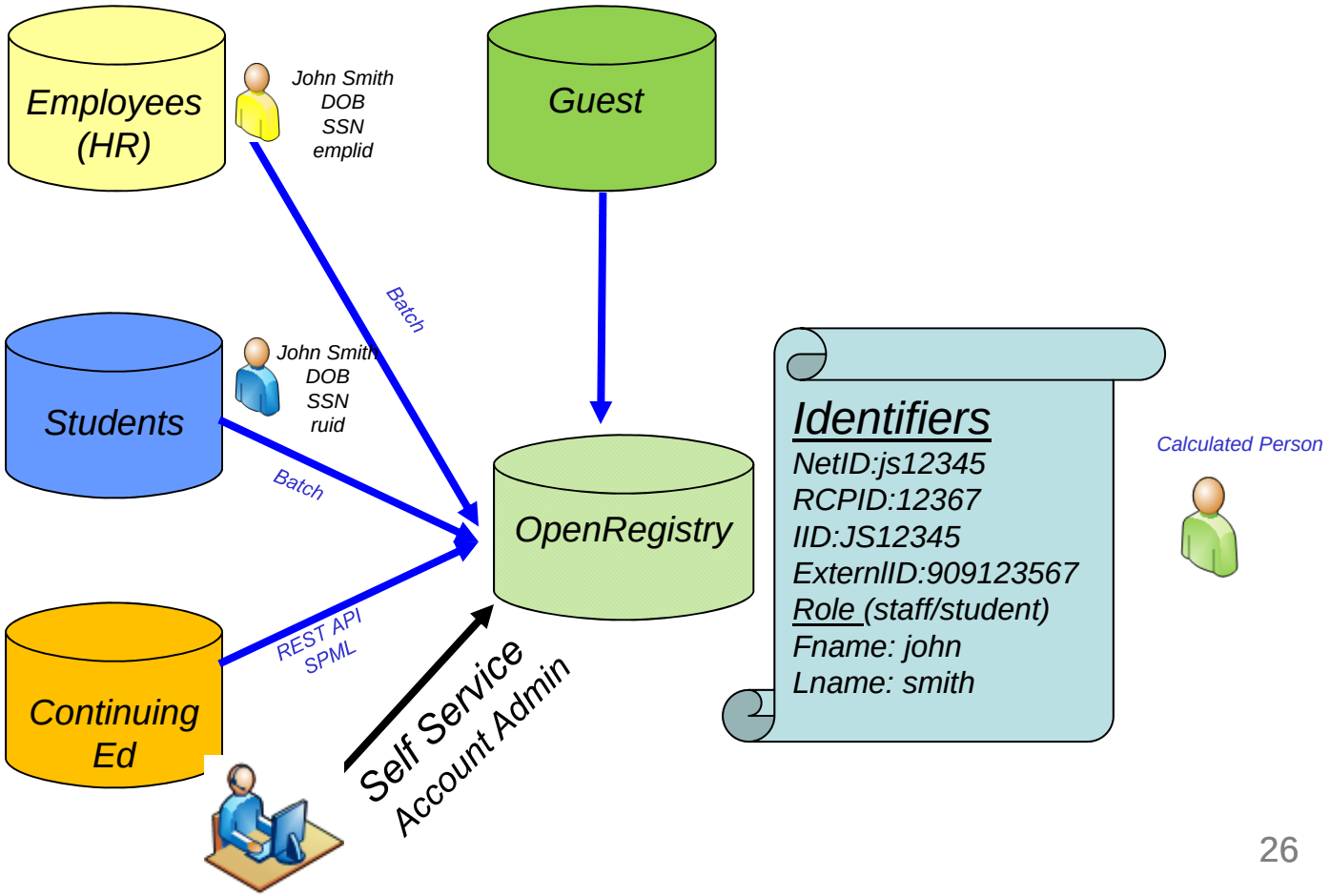
Component Architecture (Rutgers)

- **Calculate:** Calculation is an Attribute Calculation Engine, which finalizes the data elements and store them in the **c Tables.** For example , calculating users' disclosure default setting based on user's role

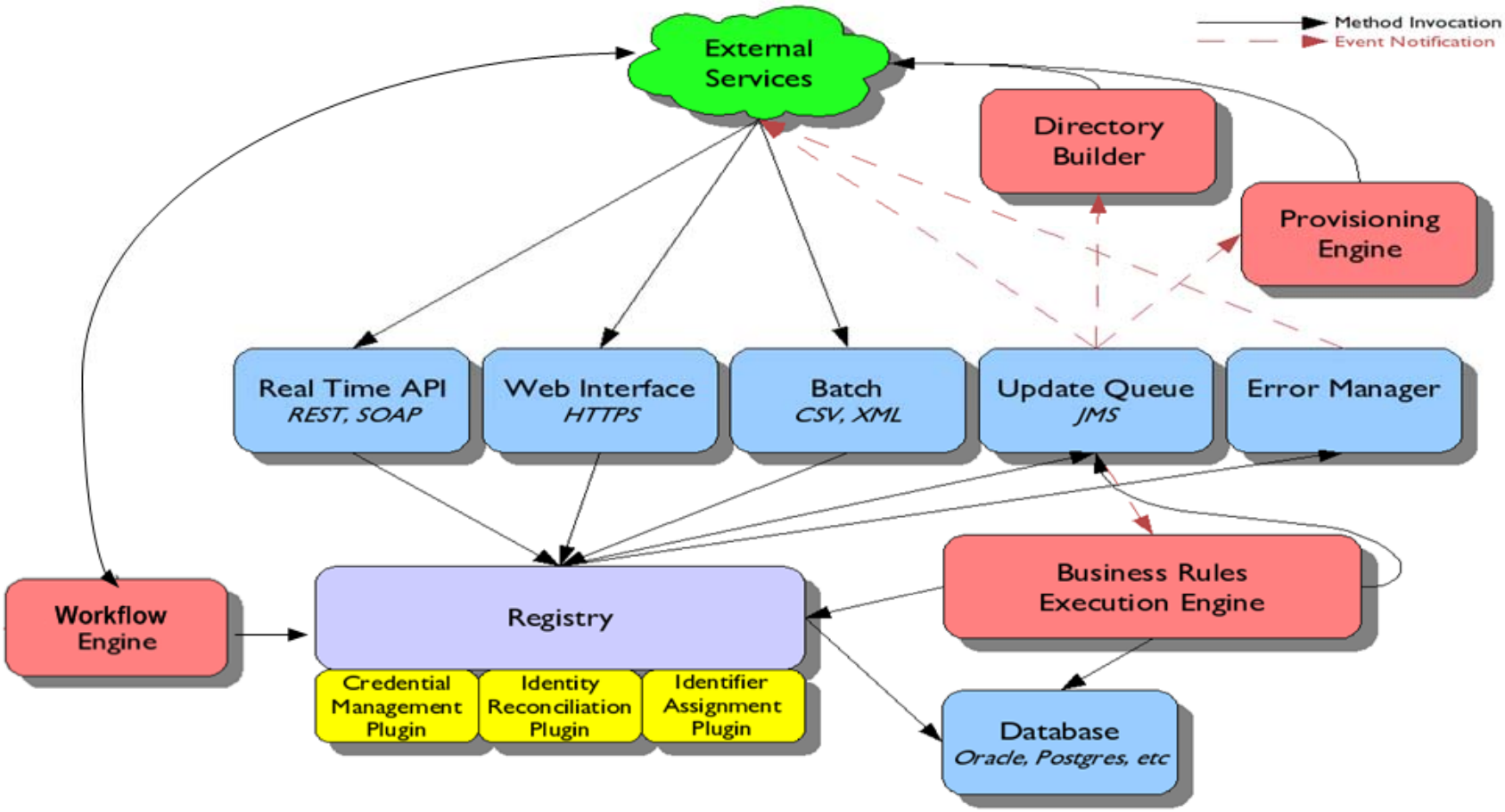
- **cTables:** *Tables used for storing Calculated "finalized" data*

- **Export:** Data is exported to Downstream Systems via batch, real-time, and provisioning interfaces . For example utilizing Camel Routing Engine to update Messaging queues ,data base tables , LDAP or flat files

Component Architecture (Rutgers)



Component Architecture



Component Architecture (Rutgers)

- **Inbound Interfaces**
  - Batch : The batch component is responsible for processing data feeds from the System of Records based on fix schedules. Data is aggregated for processing in staging tables, which then in turns call the OpenRegistry core service layers for processing
  - Open Registry provides RESTful HTTP APIs to programmatically access significant resources of the system and change their server-side state. The APIs are simple HTTP resources which conform to the basic REST (REpresentational State Transfer) principals i.e. the exposed resources are identified by unique URIs and are conformed to the 'uniform interface' i.e. basic set of the standard HTTP methods such as GET, PUT, POST, DELETE

Component Architecture (Rutgers)

- **Inbound Interfaces**
  - Available RESTful APIs (*https://wiki.jasig.org/display/ORUM/RESTful+API*)
    - Activation Key: Represents an activation key, which can be verified, invalidated, or created via a RESTful API
    - Calculated Person Resource: Allows a user to manipulate a person's system of record via a real-time API
    - Email Resource: Allows a client to update or add an email attached to a particular person's system's of record role's affiliation
    - NetID Management Resource: Allows a system to manipulate a person's netIds via a real-time API
    - SoR Person Resource: Allows a system to manipulate a person's system of record record via a real-time API
    - SoR Role Resource: Allows a system to manipulate a person's system of record role record via a real-time API
    - Non-Person Resource: Allows a system to assign NetID's to a non-perosn entity

Component Architecture (Rutgers)

- **Inbound Interfaces**
  - Web UI (SoR Data) : Integrating an application directly with OpenRegistry's Libraries. For example, OpenRegistry User Management Tools. These tools offers:
    - Create and manage of Guest accounts
    - Process approval (Used in processing early faculty staff request)
    - Update on behalf of SoR (limited to the super admin)
    - Help Desk Tool
    - View Person
    - Generate activation keys for users
    - Split/Join operation (limited to the super admin)

- **Security**
  - Spring Security is being used by OpenRegistry to secure resources. Local deployments should modify the original authentication mechanism deployed with OpenRegistry to fit their needs

Component Architecture (Rutgers)

- **Directory Builder**: OpenRegistry can facilitate the building of On-Line Directory service. The Directory builder utilizes updates from the OpenRegistry event-driven messaging service and keep up-to-date information about users

- **Provisioning Engine (Road Map): :** OpenRegistry can facilitate the building of a provisioning engine that can consume updates from the OpenRegistry event-driven routing/messages service and create accounts in the underlying downstream systems

- **Business Rules Execution Engine (Road Map):** . A Rule engine can be integrated with the OpenRegistry to validate business rules at runtime without the need to recompile and redeploy the OpenRegistry Software. Business rules and policies are dynamic in nature, for example, changing the criteria for offering new service based on a new role or extending the role's expiration date based on service account usage

Component Architecture (Rutgers)

- **Credential Management "Plug-in":** OpenRegistry stores activation keys that can be utilized for Identity activation and password management
- **Workflow Engine (Roa Map):** : Is Multiple processes/tasks connected to control identity life cycle and management . For examples:
  - Control creation, deletion, enabling and disabling of user accounts
  - Sending the activation key via email
  - Approval handling: Email for sponsors to extend the role of a guest account or to request a sponsor approval to allow creating a new guest account
  - Send messages to Resolvers to fix data anomalies that are captured during identity reconciliation (effective error handling during provisioning)
  - Improve account attestation by sending messages via email to certify that the user who needs certain access privilege is in deed qualified
  - Can be scheduled : Send notification 60 days before Guest role expires to sponsors

Component Architecture (Rutgers)

- **Outbound Interfaces**
  - Camel Routing Engine: Providing a framework to *asynchronously* , in real time, update the various backend resources (DB, LDAP, Web Services, Flat files, Messaging queues). The Routing engines captures the following events:
    - *Creating new persons*
    - *Creating new non-person entities (Program accounts)*
    - *Updating existing persons*
    - *Updating specific identifiers such as NETID*
    - *Updating roles information such as updating email address on a role*
    - *Updating disclosure information*
    - *Closing a role*

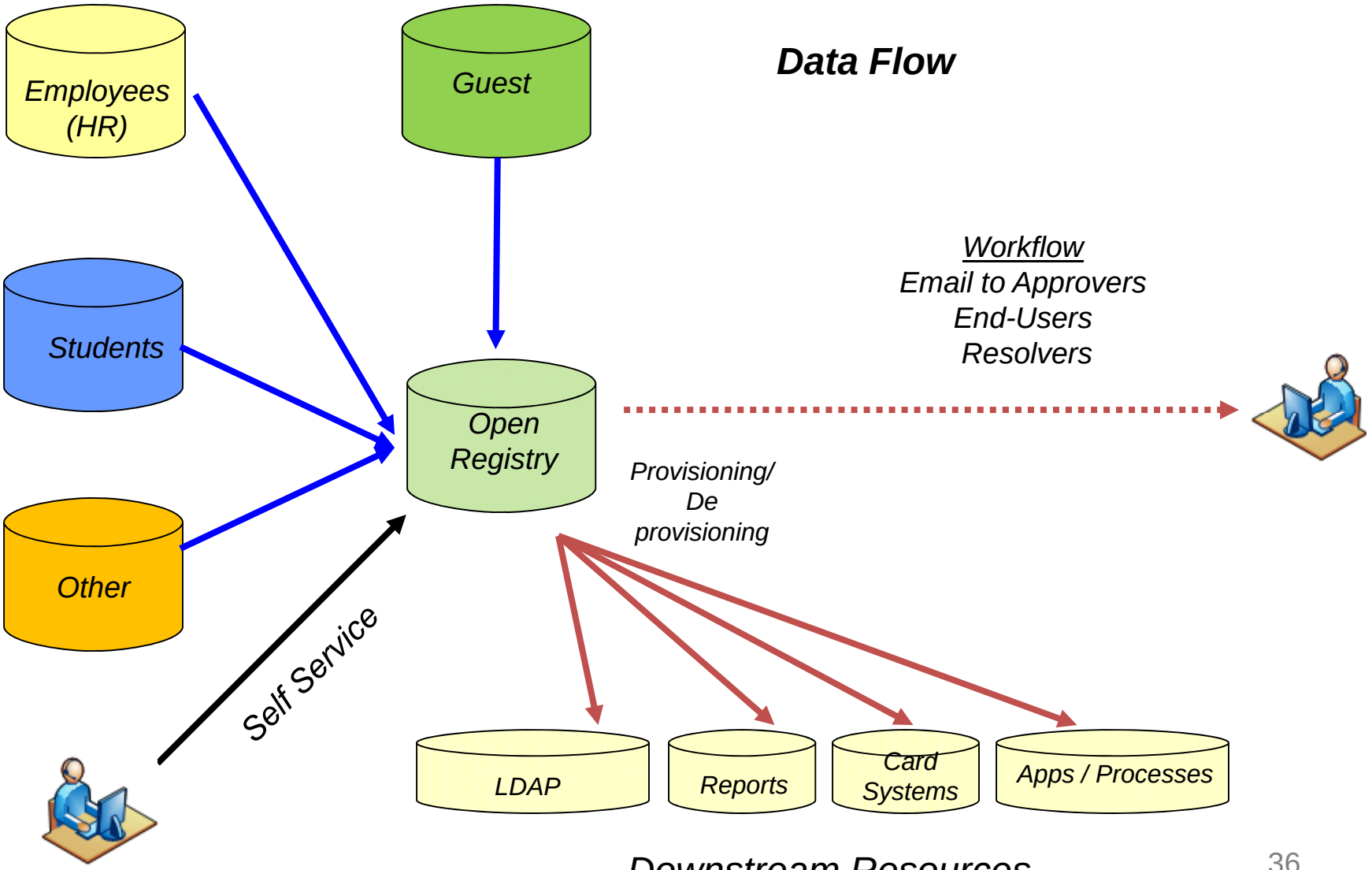
Component Architecture (Rutgers)

- **Outbound Interfaces**
  - Message Queue (Still as proof of concept) : Providing an event driven middleware (real/near time) to send messages to the downstream systems about changes in the OpenRegistry. Messages contain data elements changes. This middleware can serve as a provisioning engine via customized connectors that consume the messages and update the downstream system in near real time. For example, we can create Sakai's connectors that consume data from the OR Message Queue and directly update Sakai DB (Future Implementation)
  - Batch : Providing OpenRegistry data in bulk (fixed schedule) to downstream systems. This interface directly reads data from OpenRegistry tables and export it in a simple form
  - Report Server : Providing reports about OpenRegistry . For example Number of new users, new roles , approvals ,etc..

Component Architecture (Rutgers)

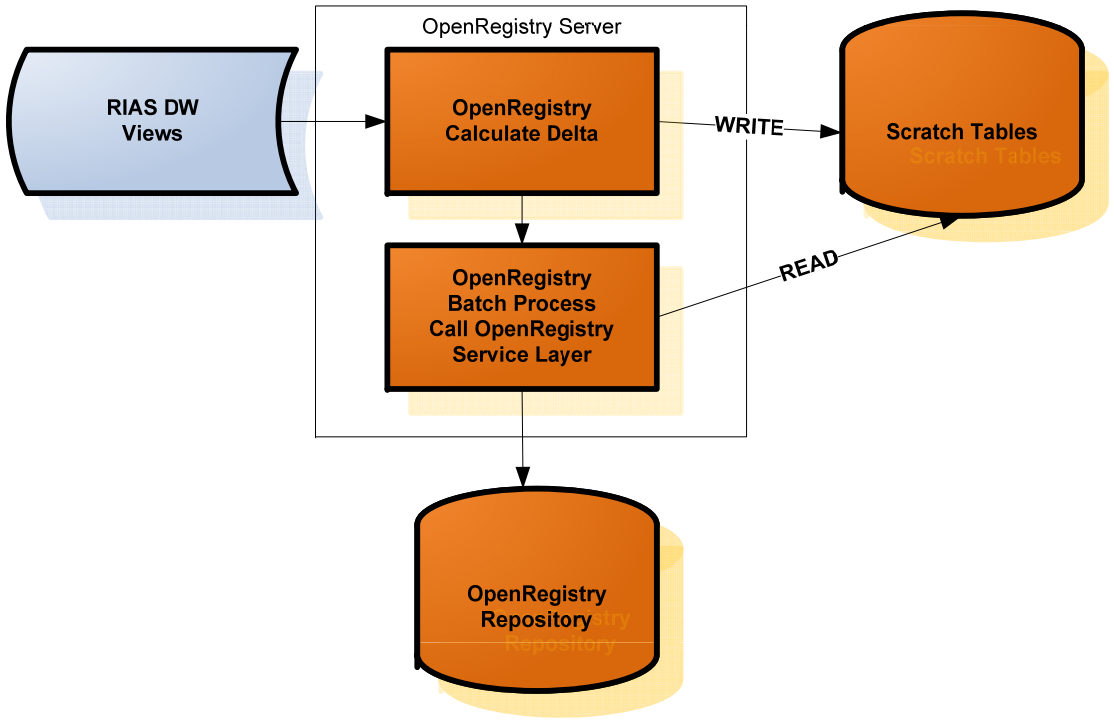
- **Outbound Interfaces**
  - Views: Simplify the OpenRegistry data model by exposing the relevant users' tables from the OpenRegistry to the downstream systems

Component Architecture (Rutgers)



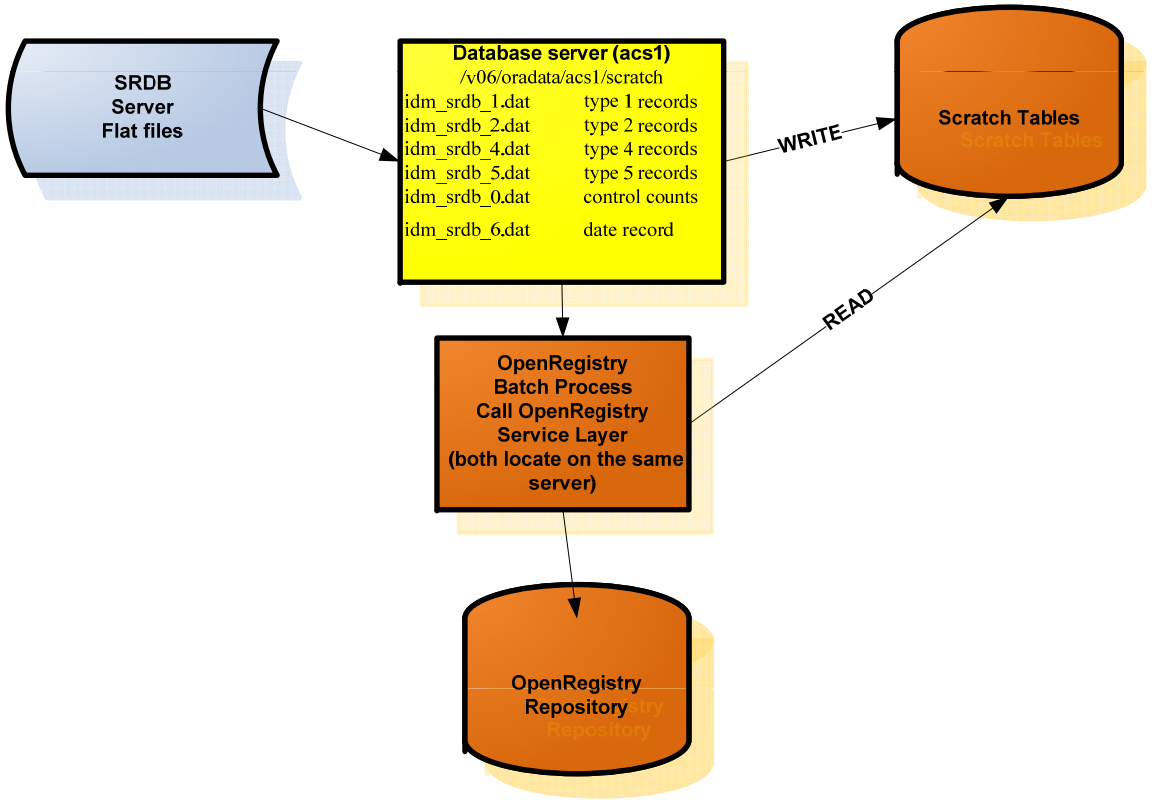
Component Architecture (Rutgers)

Inbound : *Batch Interface for Employees data feed*



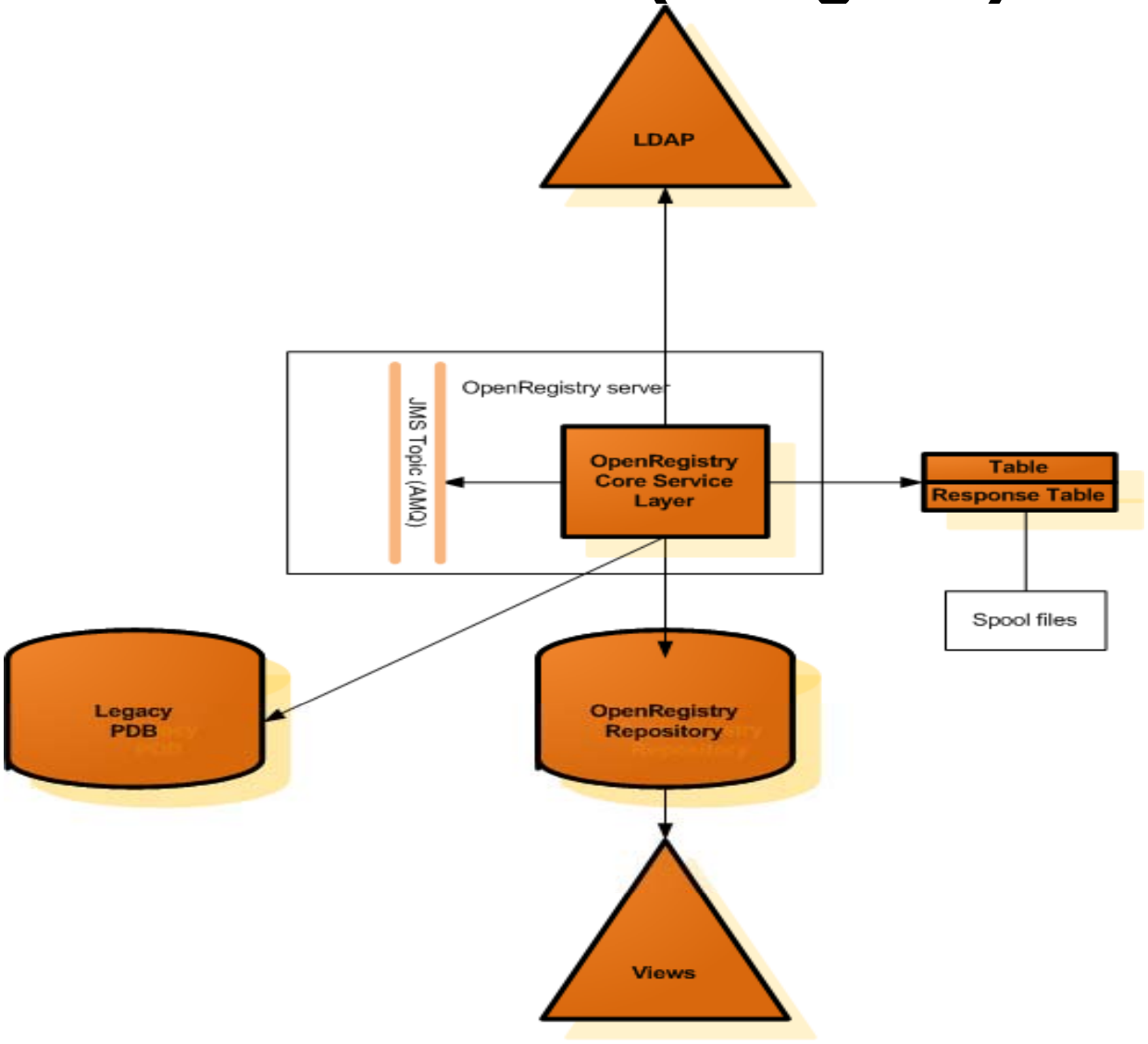
Component Architecture (Rutgers)

Inbound : *Batch Interface for Students data feed*



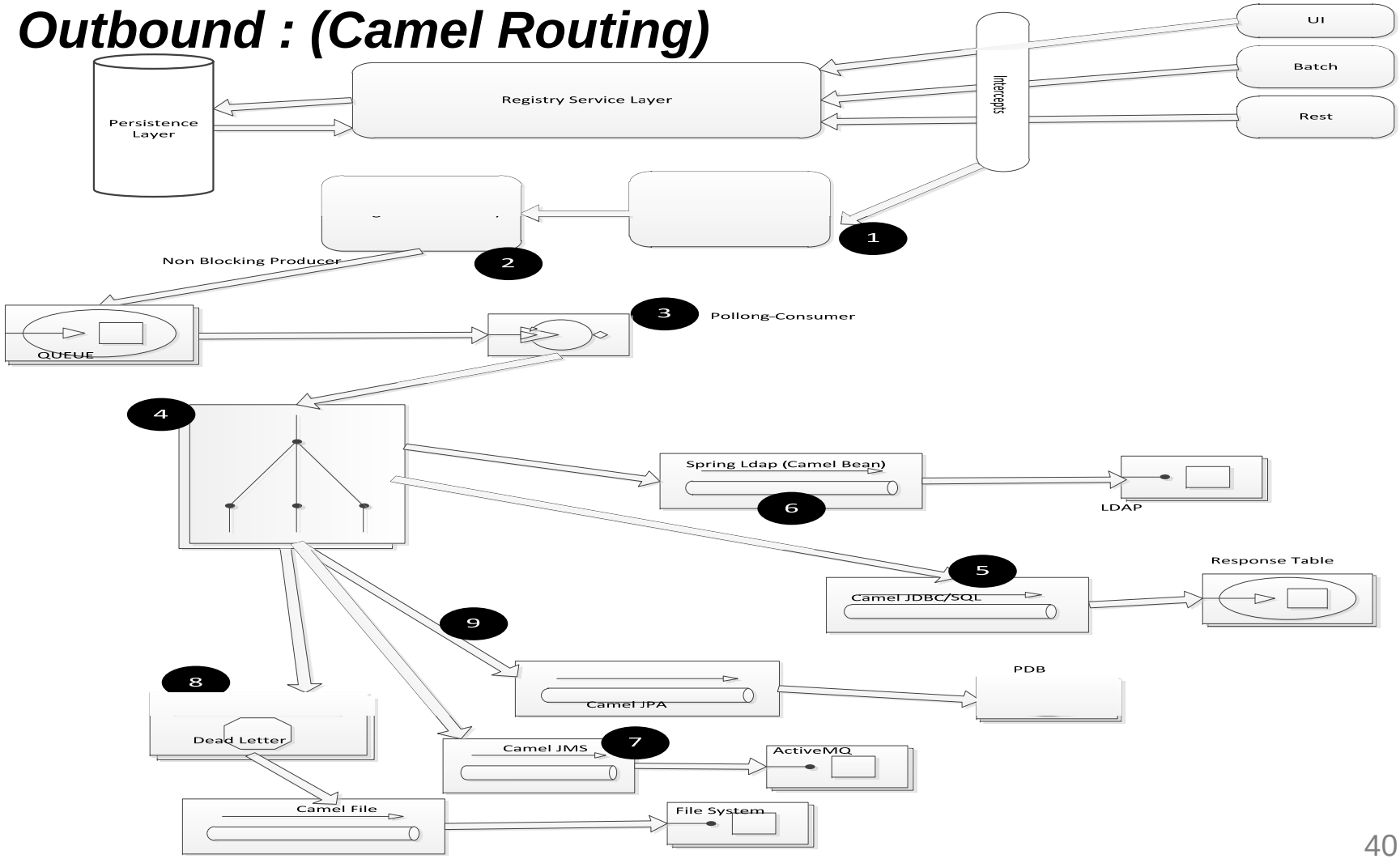
Component Architecture (Rutgers)

Outbound :



Component Architecture (Rutgers)

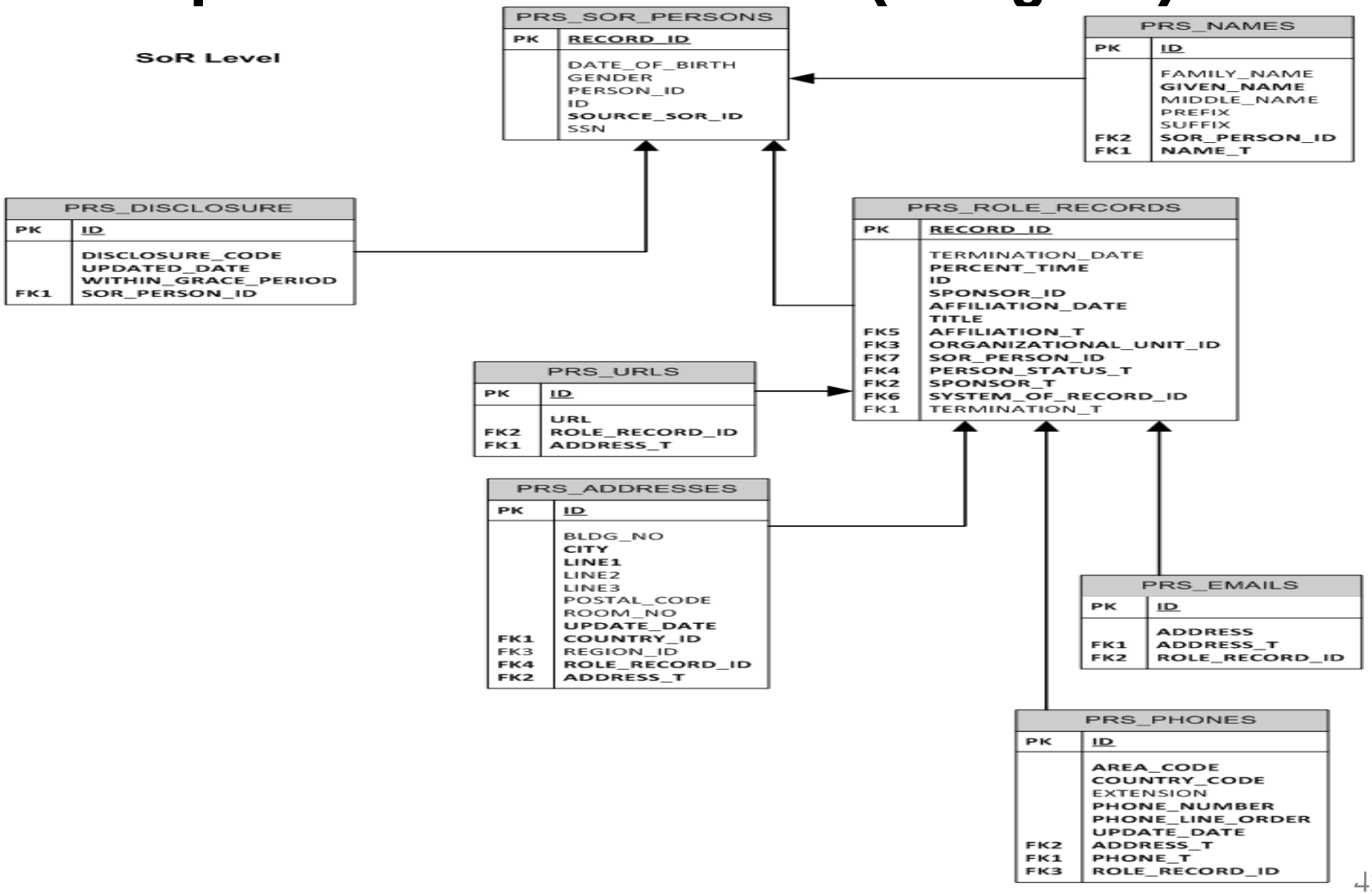
Outbound : (Camel Routing)



Component Architecture (Rutgers)

- **Data Model**
  - https://wiki.jasig.org/display/OR/Data+Model (complete data model)
  - SoR Level
    - Person
    - Roles
    - Address
    - Phone
    - Disclosure

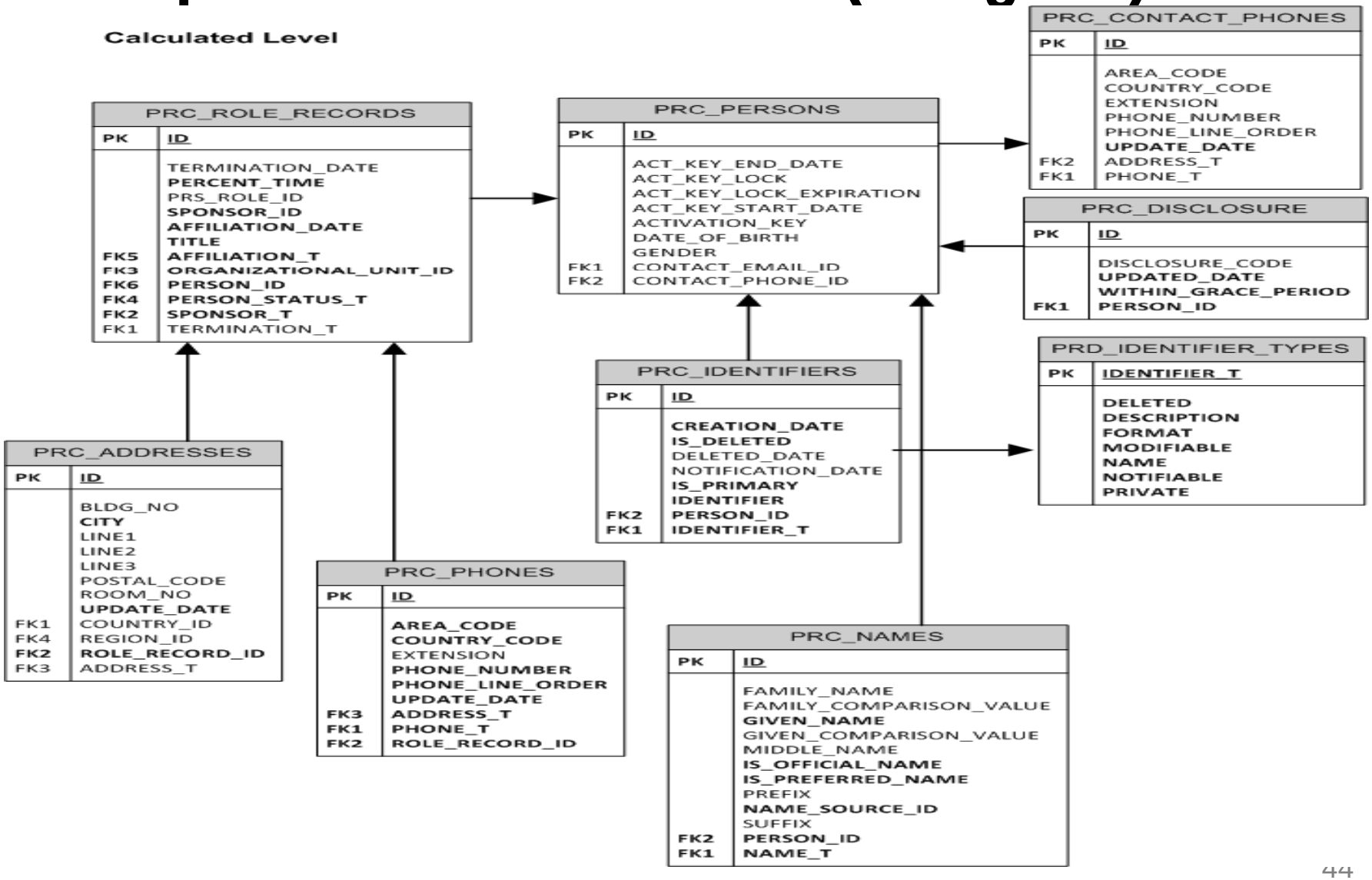
Component Architecture (Rutgers)



Component Architecture (Rutgers)

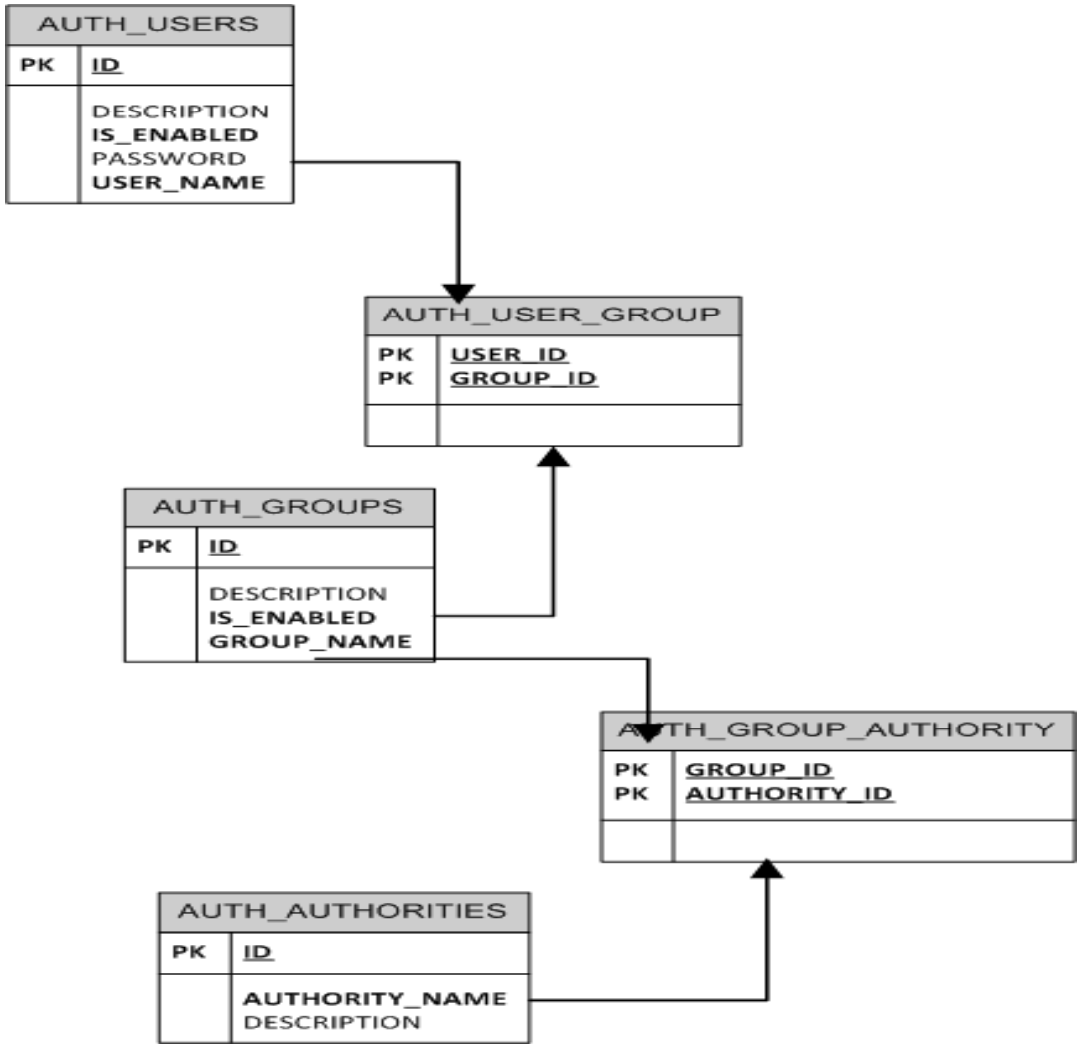
- **Data Model**
  - Calculated Level
    - Person
    - Roles
    - Address
    - Phone
    - Disclosure

Component Architecture (Rutgers)



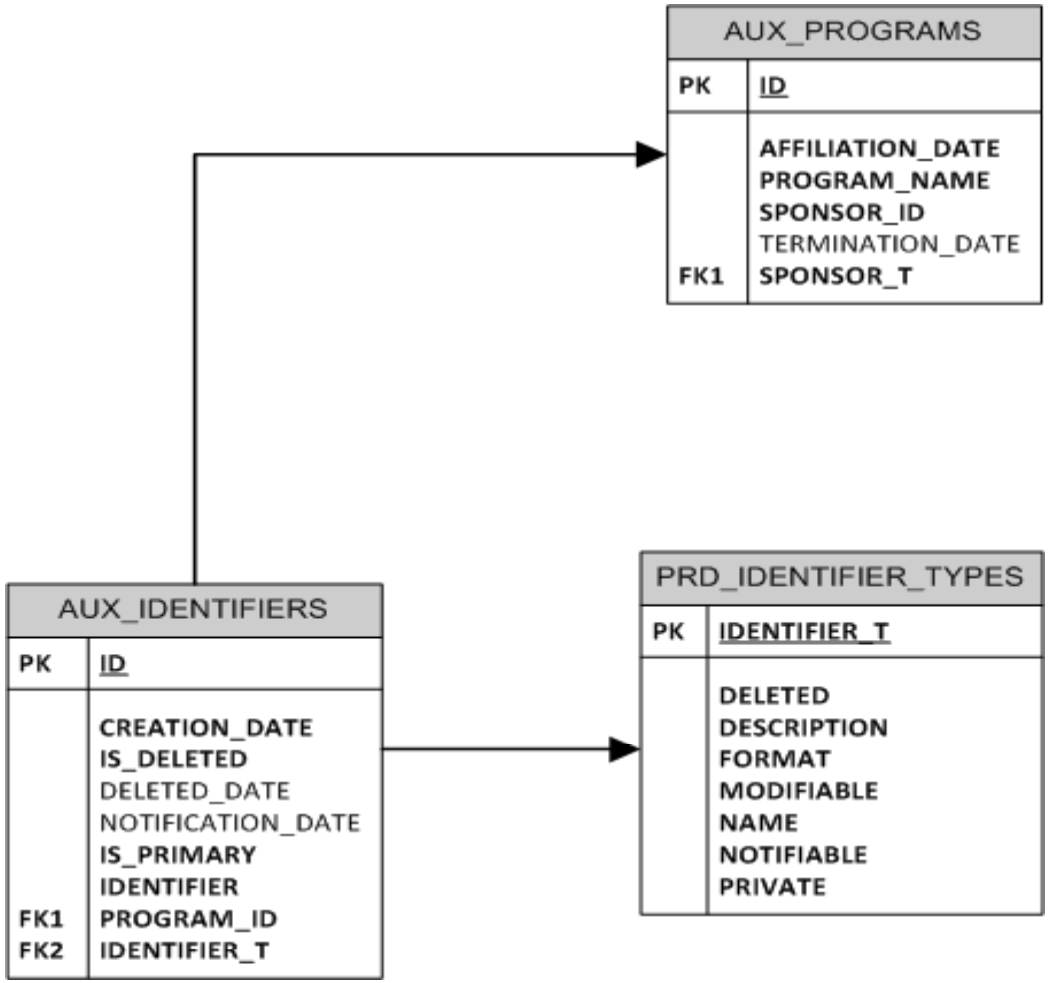
Component Architecture (Rutgers)

- **Data Model**
  - Authorization
    - Groups
    - Authorities
    - Users



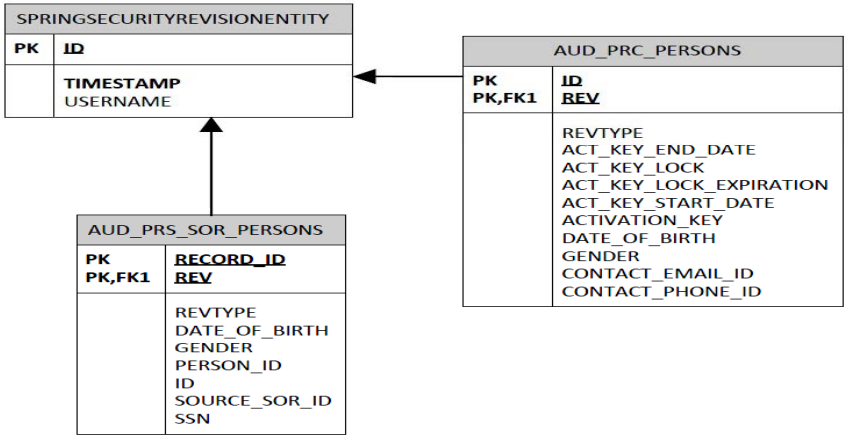
Component Architecture (Rutgers)

- **Data Model**
  - Non-person entities
    - Aux_Programs
    - AUX_identities



Component Architecture (Rutgers)

- **Data Model**
  - Audit
    - All tables follows the same



Component Architecture (Rutgers)

- **Data Model**
  - Reference tables
    - Countries
    - Regions
    - Data Structure
    - Organizational Unit

PART 3

Demo

PART 4

Code Review

Code Review

- **Election**
  - https://source.jasig.org/openregistry/trunk/openregistry-api/src/main/java/org/openregistry/core/service/FieldElector.java
  - https://source.jasig.org/openregistry/trunk/openregistry-api/src/main/java/org/openregistry/core/service/SorRoleElector.java

All elector are configured the same way, below is the example for dob elector

```
<bean class="edu.rutgers.openregistry.core.service.DateOfBirthElector" id="birthDateFieldElector">
 <property name="rolePriority">
  <list>
   <value>FACULTY</value>
   <value>STAFF</value>
   <value>FOUNDATION</value>
  <value>STUDENT</value>
  <value>ADMIT COMING</value>
  <value>SUMMER STUDENT</value>
  <value>WINTER STUDENT</value>
  <value>STUDENT WORKER</value>
  <value>GUEST</value>
  <value>ALUMNI</value>
  </list>
 </property>
</bean>
```

Code Review

- **Person Service**
  - https://source.jasig.org/openregistry/trunk/openregistry-api/src/main/java/org/openregistry/core/service/PersonService.java
- **Identifier Assigned**
  - https://source.jasig.org/openregistry/trunk/openregistry-api/src/main/java/org/openregistry/core/service/identifier/IdentifierAssigner.java
- **Reconciler**
  - https://source.jasig.org/openregistry/trunk/openregistry-api/src/main/java/org/openregistry/core/service/reconciliation/Reconciler.java

Code Review

- **Name Aspects (normalize)**
  - https://source.jasig.org/openregistry/trunk/openregistry-repository-jpa-impl/src/main/java/org/openregistry/core/aspect/LastNameAspect.java
  - https://source.jasig.org/openregistry/trunk/openregistry-repository-jpa-impl/src/main/java/org/openregistry/core/aspect/FirstNameAspect.java

Code Review (Normalization)

```
  ......
   <bean name="capitalizationAspect"
         class="org.openregistry.core.aspect.CapitalizationAspect"
         factory-method="aspectOf">
       <property name="defaultCapitalization" value="CAPITALIZE" />
       <property name="overrideCapitalization">
          <map>
             <entry key="name.given" value="NONE" />
             <entry key="name.family" value="NONE" />
          </map>
       </property>
   </bean>

   <bean name="firstNameAspect"
         class="org.openregistry.core.aspect.FirstNameAspect"
         factory-method="aspectOf">
       <property name="disabled" value="false" />
   </bean>

   <bean name="lastNameAspect"
         class="org.openregistry.core.aspect.LastNameAspect"
         factory-method="aspectOf">
       <property name="disabled" value="false" />
   </bean>
</beans>
```

Code Review

- **Camel Routing**
  - Aop is being used in OpenRegistry to capture the events (add ,update). Aspects then invoke camel to log events in database table.

Route for Camel Producer:

```
<route id="person-event_queue" shutdownRoute="Defer">
   <from uri="seda:person-event?concurrentConsumers=1"/>
   <to uri="jpa:edu.rutgers.pdb.domain.IdmEventQueue"/>
</route>
```

Code Review

- **Camel Routing**

Route for Camel Consumer:

```
<!-- Event message flow entry point -->
      <route id="person-event-processor" shutdownRoute="Defer">
         <from uri="direct:person-event-queue-processor"/>
         <choice>
            <when>
               <simple>${in.body.eventType} == 'PERSON_CREATED_EVENT'</simple>
                <to uri="direct:person-created"/>
            </when>
             ……..……
            <when>
               <simple>${in.body.eventType} == 'NETID_CHANGED'</simple>
               <to uri="direct:person-updated"/>
            </when>
            <when>
               <simple>${in.body.eventType} == 'NONPRIMARY_NETID_ADDED'</simple>
               <to uri="direct:person-updated"/>
            </when>
             <otherwise>
               <throwException ref="unknownEventTypeException"/>
            </otherwise>
         </choice>
      </route>
```

Code Review

- **RestFul API's**
  - *https://wiki.jasig.org/display/ORUM/RESTful+API*

- **Authentication / Authorization**
  - *https://wiki.jasig.org/display/ORUM/Authentication+and+Authorization*

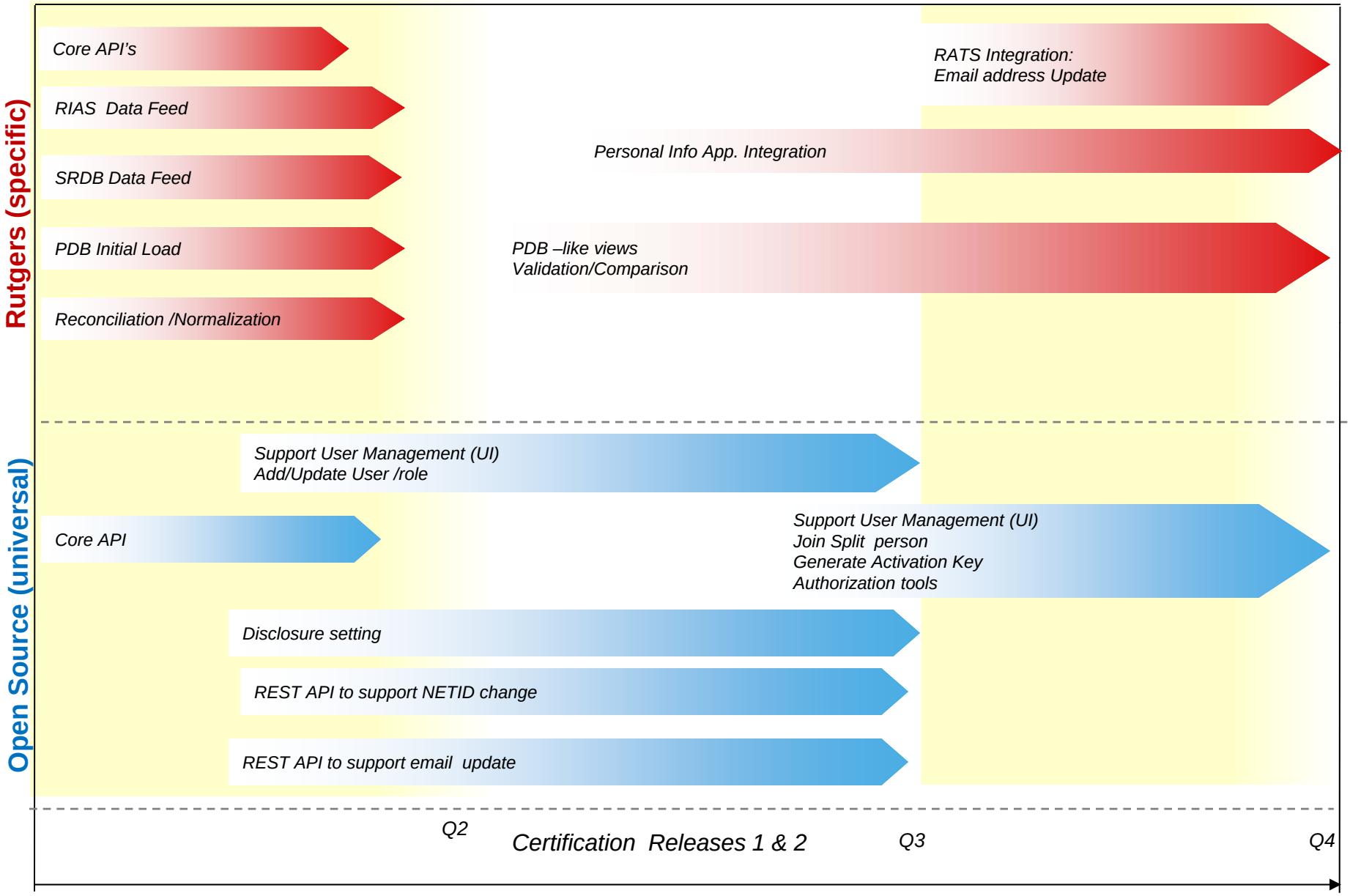
Code Review

- **Authentication/Authorization**

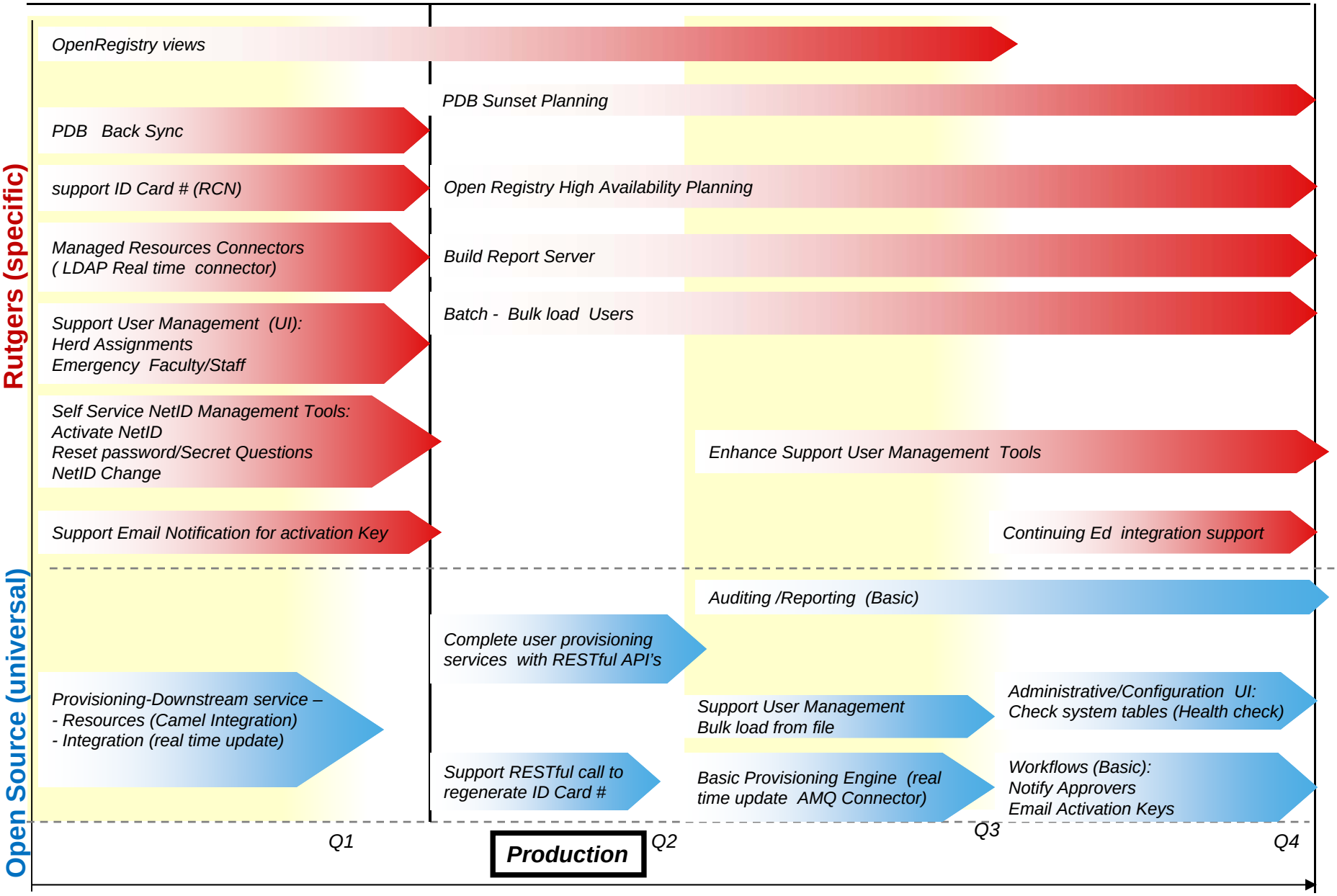
```
bean id="filterInvocationInterceptor"
    class="org.springframework.security.web.access.intercept.FilterSecurityInterceptor"
     p:authenticationManager-ref="authenticationManager"
     p:accessDecisionManager-ref="accessDecisionManager">
     <property name="securityMetadataSource">
        <sec:filter-security-metadata-source path-type="ant" use-expressions="true">
          <!-- securing email resource-->

 <sec:intercept-url pattern="/**/email*"  access="hasAnyRole('ROLE_EMAIL_READ','ROLE_ADMIN')"
    method="GET"/>
 <sec:intercept-urlpattern="/**/email*“access="hasAnyRole('ROLE_EMAIL_UPDATE','ROLE_ADMIN')"
    method="POST"/>
<sec:intercept-url pattern="/**/email/SoRs*"  access="hasAnyRole('ROLE_EMAIL_UPDATE','ROLE_ADMIN')"
    method="POST"/>
<sec:intercept-url pattern="/**/activation/*"  access="hasAnyRole('ROLE_GENERATE_KEY','ROLE_ADMIN')"
    method="DELETE"/>
       ……..
       ……..
<sec:intercept-url pattern="/addPerson.htm"  access="hasRole('ROLE_ADD_PERSON')and
    hasRole('ROLE_ADD_SOR_ROLE') or hasRole('ROLE_ADMIN')"/>
 </sec:filter-security-metadata-source>
     </property>
  </bean>
```

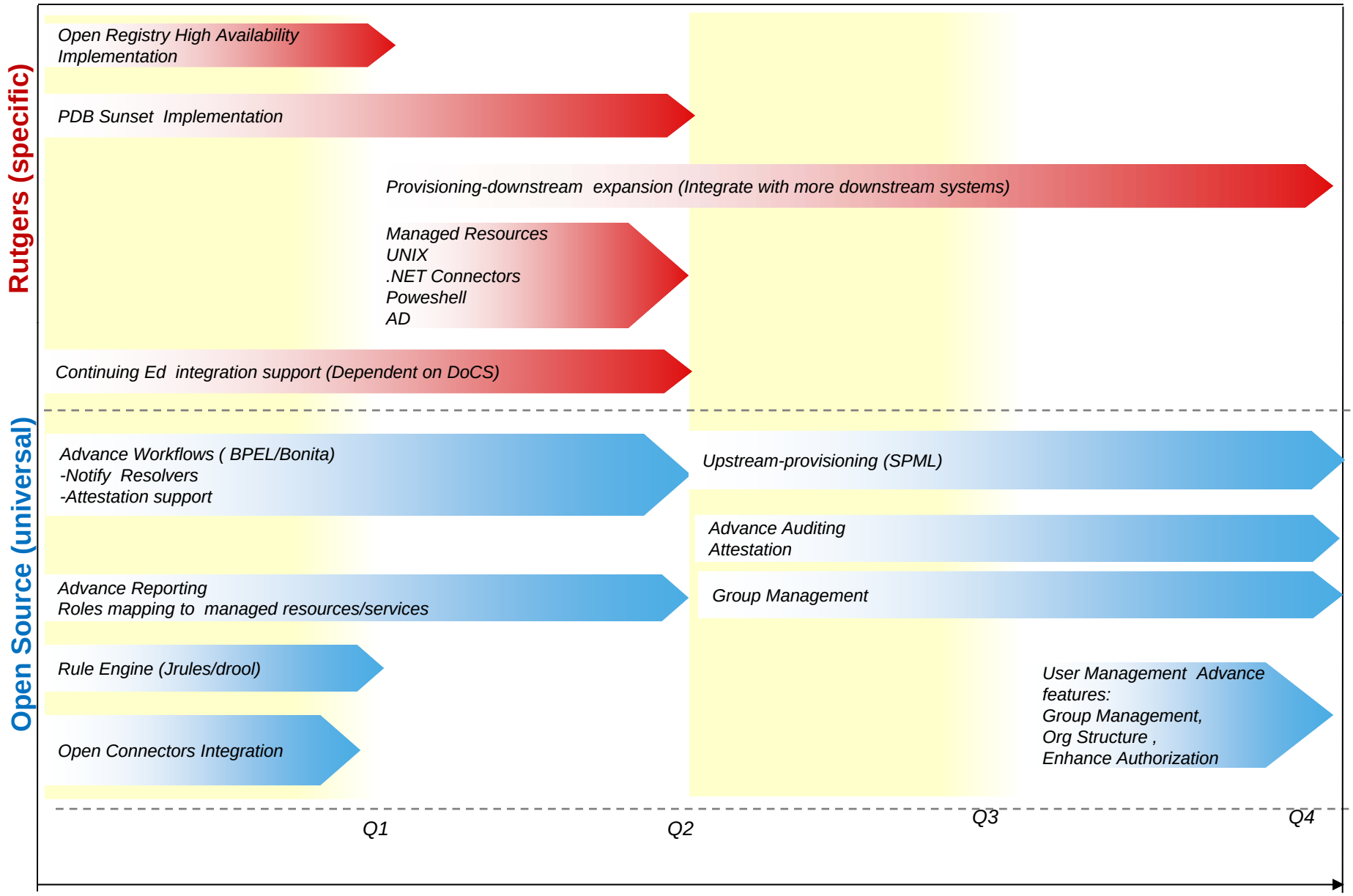
RUTGERS OpenRegistry – Road Map 2011 COMPLETED



RUTGERS OpenRegistry – Road Map 2012



RUTGERS OpenRegistry– Road Map 2013



Miscellaneous

- Rutgers Implementation:
  - SpringFramework 3.0.5
  - Hibernate.version 3.5.5
  - Camel version 2.5.0
  - ActiveMQ version 5.4.2
  - Tomcast version 6.0.32
  - Java SDK 1.6
  - Database: Oracle 11g
  - HW: Sun

Miscellaneous

- Source Code:
  - https://source.jasig.org/
  - https://source.jasig.org/openregistry/trunk/
- Home
  - https://wiki.jasig.org/display/OR/Home
- Design/High Level architecture
  - https://wiki.jasig.org/display/OR/High+Level+Architecture
- Data Model
  - https://wiki.jasig.org/download/attachments/17006634/riar_db_model_v2_28_2012.pdf?version=1&modificationDate=1330463253883
- Development
  - https://wiki.jasig.org/display/JSG/openregistry-dev

Miscellaneous

- User Manual
  - https://wiki.jasig.org/display/ORUM/Home

THANK YOU

Omer Almatary
IdM Project Manager, ESS
oalmatar@rutgers.edu
Muhammad Siddique
Application Architect/Developer
msidd@rutgers.edu