



## **KS Curriculum Management 2.0 Release Notes**

KS Curriculum Management 2.0 Source Code .....	6
KS War Name Changes and Deployment Strategy Notes .....	7

# KS Curriculum Management 2.0 Release Notes

## In This Document

1. What's New
2. What You Can Expect To Do with this Release
3. Bug Fixes Included in Release 2.0
4. Bug Fixes included in Release 2.0.1  
CM 2.0.1 Fixed Issues (10 issues)
5. Bug Fixes included in Release 2.0.2  
CM 2.0.2 Fixed Issues (20 issues)
6. Source Code

(For support and maintenance options please refer to the [Kuali Student Support and Maintenance Guide](#))

## 1. What's New

### New Feature - Retire Course by Proposal

1. An approved course may be retired through the proposal process, indicating important information like the last term in which the course can be offered and retirement rationale.
2. Curriculum managers can be authorized to access a special administrative screen to retire a course. Using this streamlined screen, the curriculum manager can retire a course and is not required to go through the standard review and approval process.

### Component Updates

KS CM 2.0 released with updated service contracts and versions of Rice and GWT.

1. Rice has been updated from v1.0.3 to v2.2
2. GWT has been updated from v2.1 to v2.3
3. Service Contracts have been updated from v1.0 to v2.0

### KS War Name Changes:

To accurately reflect the purpose of the builds and align Kuali Student with Rice naming conventions, Kuali Student has renamed the builds.

The war artifacts have been renamed as follows:

Original war artifact (CM1.x)	New war artifact (CM2.x)
ks-embedded	ks-with-rice-bundled
ks-standalone	ks-with-rice-embedded
ks-rice	ks-rice-standalone

The database directories have been renamed as follows:

Original database directory (CM1.x)	New database directory (CM2.x)
ks-embedded-db	ks-bundled-db
ks-standalone-db	ks-app-db
ks-rice-db	(no change)

For more information about name changes and naming conventions, see [KS War Name Changes and Deployment Strategy Notes](#).

## 2. What You Can Expect To Do with this Release

### 2.1 Before You Start

Before you install Kuali Student, please do the following to familiarize yourself with the system:

- Read the available documentation [CM 2.0 Documentation](#).
- Review the [Service Contracts](#) which include service descriptions and message data structures

## 2.2 Install Kuali Student

See the [KS 2.0 Installation Instructions](#) for detailed instructions.

## 2.3 Upgrade Database from 2.0.0 to 2.0.1

 Only apply this schema fix script if your upgrade path stops at 2.0.1. For upgrade to 2.0.2, this script should be skipped as its contents are included in the database upgrade scripts for 2.0.2.

If you have a previous installation of 2.0 run this script against your database to resolve a known issue with generated constraint names: [constraint\\_fix.sql](#)

## 2.4 Upgrade Database from 2.0.1 to 2.0.2

With Curriculum Management 2.0.2, the following database changes were introduced:

- Unnecessary tables were dropped from the `KSBUDED` and `KSAPP` schemas.
- Auto generated constraint names were renamed to correspond to Curriculum Management naming standards.
- Hard coded text from some GWT screens was moved from application to database message catalog.

You can upgrade your CM 2.0.1 database to CM 2.0.2 by executing the following scripts. It is strongly recommended that the upgrade scripts be inspected and tested on a non-production system before applying to production.

1. `upgrade-db-2.0.1-to-2.0.2-drop-extra-tables.sql`
2. `ks-lum-2.0.2-cm/ks-lum-sql/src/main/resources/upgrades/18-CM-2.0.2/2013-09-12-message-catalogue.sql`
3. `ks-lum-2.0.2-cm/ks-lum-sql/src/main/resources/upgrades/18-CM-2.0.2/2013-09-24-program-message-catalogue.sql`
4. `ks-core-2.0.2-cm/ks-core-sql/src/main/resources/upgrades/18-CM-2.0.2/2013-10-11-fix-core-constraint-names.sql` (skip if `constraint_fix.sql` has already been applied from previous step)
5. `ks-lum-2.0.2-cm/ks-lum-sql/src/main/resources/upgrades/18-CM-2.0.2/2013-10-11-fix-lum-constraint-names.sql` (skip if `constraint_fix.sql` has already been applied from previous step)

## 3. Bug Fixes Included in Release 2.0

Key	Summary
KSENROLL-4257	CM: Node Name Null Exception when Returning Doc to Previous Node
KSENROLL-4252	CM: Learning Objectives are not being saved in course proposal
KSENROLL-4205	CM: Saving Requisites fails
KSENROLL-4087	CM: Validation Error when submitting proposal with Multiple Credit Value (s)
KSENROLL-4083	CM: WorkflowServiceErrorException thrown when Proposer withdraws their own course proposal
KSENROLL-4079	CM: Missing correct "Proposal Actions" for course proposal enroute
KSENROLL-4042	CM: Program Rationale not being saved; prevents Blanket Approval
KSENROLL-3882	CM: ks-distributed: Searching by Key Word for Learning Objective wipes out Course Information
KSENROLL-3866	CM: ks-distributed: Adding Instructor wipes out Course Information
KSENROLL-3734	CM: Program Proposal falls into exception: "id to load is required for loading"

KSENROLL-3718	CM ks-distributed: The first person on the workflow that is supposed to approve a submitted proposal can not edit (add) Activity Type
KSENROLL-3675	CM: Submitting Course Proposal to Workflow fails - creditOptions Validation Error
KSENROLL-3605	CM ks-distributed: When user eric creates a blank course proposal, the proper workflow nodes are not created and the doc goes directly to final state
KSENROLL-3596	CM: Submit course proposal to workflow fails- Failed to locate a document type with the given name
KSENROLL-3579	CM: Statement service exception when picking Prerequisite Rules
KSENROLL-3578	CM: Save Failed when adding a Version Code to a Course Proposal.
KSENROLL-3577	CM: Save Failed when adding a Jointly Offered Course to a Course Proposal.
KSENROLL-3575	CM: Save Failed when adding a Cross Listed Course to a Course Proposal.
KSENROLL-3400	CM Start Term disappears when clicking "Approve and Activate" a saved course proposal as admin user
KSENROLL-3382	KIM Permission issue in distributed setup causing recursive login loop

Showing 20 out of 96 issues

## 4. Bug Fixes included in Release 2.0.1

### CM 2.0.1 Fixed Issues (10 issues)

Key	Summary
KSCM-724	CreateOrg() method fails to add the new Org into the database
KSCM-725	StudentSpringBeans.xml is still creating a Document Service even though Rice 2.0 evidently does so
KSCM-726	Two versions of Dozer mapping file with different contents can break application
KSCM-737	Search in SearchableCrudDaoImpl is only processing first state in list
KSCM-738	LRC Service Name incorrect
KSCM-753	find course should display a course
KSCM-763	Improper column used in order by for EnumeratedValueDao lookup SQL
KSCM-765	Grading options are not properly disassembled
KSCM-840	version bump for kuali-common to allow publishing of artifacts to maven central
KSCM-841	avoid use of jdk7 for compilation of curriculum management

10 issues

## 5. Bug Fixes included in Release 2.0.2

### CM 2.0.2 Fixed Issues (20 issues)

Key	Summary
KSCM-843	Remove Unused Entities from CLU Service
KSCM-850	Bug- Dependency Analysis hangs when trying to load a course that has statements connected to a deleted clu
KSCM-852	Code in CopyCourseServiceImpl is swallowing exception
KSCM-960	Recent modification of EnumerationManagementServiceImpl.search breaks backwards compatibility
KSCM-1004	Cannot retire by proposal when collaborators are added with 'view' permission
KSCM-1033	When creating a new ResultValueGroup, the CourseAssembler does not copy the name and descr (RichTextInfo) properties.
KSCM-1034	UploadServlet treating file extensions in a case sensitive manner
KSCM-1040	misc GWT Changes for institutional overrides
KSCM-1042	GWT overrides for cluSet Management changes
KSCM-1115	Replace hard coded values with Message Catalogue lookups
KSCM-1116	Inconsistent use of upper and lower case for draft state in KSLU_CLU table is breaking a lookup
KSCM-1117	Refactor parametrised constructors into default constructors with an init method
KSCM-1118	Change private method access modifiers to allow for overriding in Custom project
KSCM-1119	Replace instantiation of UI classes using Java new operator with GWT.create() deferred binding
KSCM-1122	KSSelectedList throws NullPointerException when setting the hide property in the dictionary file.
KSCM-1123	ClassCastException while deleting an uploaded document
KSCM-1124	Add new metadata property to replace hard coded value in the implementation
KSCM-1127	Data validation exception is always thrown as an OperationFailedException.
KSCM-1128	Data validator causing a ClassCastException
KSCM-1133	Cannot customize program language in clu set editor widget

Showing 20 out of 26 issues

## 6. Source Code

The source code for KS Curriculum Management 2.0 releases can be download [here](#).

### KS Curriculum Management 2.0 Source Code



#### Use of svn:externals in subversion repo

Kuali Student Curriculum Management 2.0 uses `svn:externals` feature in the subversion repositories.

Even though the top-level directory appears to contain no subdirectories when browsing it over https, Subversion clients will detect the `svn:externals` definitions and checkout the mapped locations when checking out the top-level project.

The source for Kuali Student Curriculum Management is publicly available from the Kuali Foundation subversion repository.

## KS Curriculum Management 2.0 Source Code

### Curriculum Management 2.0.0

<https://svn.kuali.org/repos/student/enrollment/aggregate/tags/student-2.0.0-cm/>

### Curriculum Management 2.0.1

<https://svn.kuali.org/repos/student/enrollment/aggregate/tags/student-2.0.1-cm/>

### Curriculum Management 2.0.2

<https://svn.kuali.org/repos/student/enrollment/aggregate/tags/student-2.0.2-cm/>

## KS War Name Changes and Deployment Strategy Notes

### War Name Changes/Deployment Strategy Notes

Deployment Strategy	Explanation	Supported?	Suitability
Bundled	A deployment strategy that combines the rice server with the Kuali Student application server. There are no requirements on combining the data sources but in our case they share a schema.		For development, on a single server
Distributed	A deployment strategy where applications are deployed with Rice-Embedded and a Rice-Standalone server deployed separately. This is the currently supported production deployment strategy which is currently running in production at Maryland in a load balanced environment with CM 1.2 (4 KS Servers with Rice-Embedded and 4 Rice-Standalone servers).		For production, on 2 separate servers (or 2 clusters of servers)
Remote	A deployment strategy where application and rice servers do not have database dependencies on each other. Rice does not currently support this deployment but we're working toward this goal.		N/A



#### Rice-Embedded

A group of rice plugin components which allow applications to be deployed in a separate jvm from rice and still use rice services like KEW, KRAD, KRMS, and KIM. These components require a database connection to the rice server.



#### Rice-Standalone

a fully functioning rice server that has remote services enabled for other applications. This server has some administrative applications but for the most part is meant as a backend for other applications.

## Summary of name changes

The war artifacts have been renamed (in a verbose naming fashion) as follows:

Original war artifact (CM1.x)	New war artifact (CM2.x)
ks-embedded	ks-with-rice-bundled
ks-standalone	ks-with-rice-embedded
ks-rice	ks-rice-standalone

The database directories have been renamed as follows:

Original database directory (CM1.x)	New database directory (CM2.x)
ks-embedded-db	ks-bundled-db
ks-standalone-db	ks-app-db
ks-rice-db	(no change)

The config files have been renamed to match the war files as follows:

Original config file (CM1.x)	New config file (CM2.x)
ks-embedded-config.xml	ks-with-rice-bundled-config.xml
ks-standalone-config.xml	ks-with-rice-embedded-config.xml
ks-rice-config.xml	ks-rice-standalone-config.xml

Deployment Strategy Diagram



## **KS Curriculum Management 2.0 Functional Overview**

(CM 2.0) 1. Overview .....	3
(CM 2.0) 1.1 Summary of Features Included in KS Curriculum Management 2.0 .....	3
(CM 2.0) 1.2 Configuring KS Curriculum Management for Your Institution .....	3
(CM 2.0) 2. Course Management .....	3
(CM 2.0) 2.1 Course Information .....	3
(CM 2.0) 2.1.1 Course Requisite Rules .....	4
(CM 2.0) 2.2 Course Processes .....	4
(CM 2.0) 3. Program Management .....	4
(CM 2.0) 3.1 Program Information .....	4
(CM 2.0) 3.2 Program Processes .....	5
(CM 2.0) 4. Course Set Management .....	5
(CM 2.0) 5. Learning Objectives Management .....	5
(CM 2.0) 6. Organization, People and Roles Management .....	6
(CM 2.0) 7. Reference Implementation Configuration .....	7
(CM 2.0) Field Requiredness by Workflow Node Configuration .....	7
(CM 2.0) Organization Configuration .....	11
(CM 2.0) Subject Areas and Curriculum Org Relationship Configuration .....	12
(CM 2.0) User Configuration .....	14
(CM 2.0) Workflow Configuration and Associated Users .....	15
(CM 2.0) 8. Course and Proposal States .....	18

# KS Curriculum Management 2.0 Functional Overview

## (CM 2.0) 1. Overview

This document provides an overview of the functionality delivered in Kuali Student Curriculum Management 2.0, including course and program management. The audience for this document is primarily business analysts who need to translate an institution's business requirements of subject matter experts into configuration requirements. The [KS Curriculum Management 2.0 Configuration Guide](#) contains the details for how to change the system to meet institutional requirements, while this document provides an overview of how the Kuali Student System is configured as delivered.

## (CM 2.0) 1.1 Summary of Features Included in KS Curriculum Management 2.0

Below is a summary of the features available in KS Curriculum Management 2.0. These features are covered in more detail later in this document. For an exhaustive list of features included in KS Curriculum Management 2.0, see the [KS Curriculum Management 2.0 Feature Matrix](#).

### 1. Course Management

- a. New courses are created and existing ones modified or retired through a proposal process that allows users to collaborate on a proposal, and then submit the course for review. The system determines reviewers and the review path based on roles within the institution.
- b. Curriculum managers can use special administrative screens to create, modify, or retire courses. These streamlined screens allow users to manage the course inventory without using the standard proposal's curriculum review process.
- c. Courses may also be searched for and browsed.

### 2. Program Management

- a. Academic programs are created and modified by curriculum managers using special administrative screens. Undergraduate and graduate academic programs are defined by identifying descriptive information, requirements, learning objectives and governance and oversight.
- b. Existing programs also can be modified through a proposal process similar to course management, which supports collaboration, workflow routing, and approval.
- c. Programs may also be searched for and browsed.

### 3. Course Set Management

- a. Sets of courses can be defined and named for use in managing rules for courses (e.g. prerequisites) and programs (e.g. completion requirements).

### 4. Learning Objectives Management

- a. Learning objectives provide a way to manage the inventory of learning objectives. A learning objective is a statement that describes the knowledge, skills or abilities that a student would expect to gain by participating in a course or program. KS Curriculum Management provides tools to categorize, search, and re-use learning objectives.

### 5. Dependency Analysis

- a. Dependency analysis reports demonstrate relationships among curriculum components by showing the courses, programs, and course sets that use a selected course. This is particularly useful for understanding the potential impact of curricular changes.

### 6. Organization and Workflow

- a. Information about the people and organization within an institution is maintained through Kuali Identity Management (KIM). The roles assigned to people through their affiliation and position in the organization defines the approval path for courses and programs. Details on what roles are required in the approval process are maintained through Kuali Enterprise Workflow (KEW).

## (CM 2.0) 1.2 Configuring KS Curriculum Management for Your Institution

KS Curriculum Management can be configured for your institution's specific business processes and requirements. What information is specified for a course or program proposal, what information is required and who is involved in the approval process are fully configurable as detailed in the [KS Curriculum Management 2.0 Configuration Guide](#). A business analyst who understands the business requirements of the institution will need to work with a Java developer to implement most configuration changes.

## (CM 2.0) 2. Course Management

KS Curriculum Management provides functionality for proposing, approving, modifying, and activating courses that can then be published in a course catalog. Curriculum managers can also retire courses. What information is specified for a course is fully configurable by the institution.

## (CM 2.0) 2.1 Course Information

KS Curriculum Management provides the means to record the details of a course. You can configure Curriculum Management to present and require data fields based on your institution's specific business requirements. The following is a summary of the information that is included in KS Curriculum Management as it is delivered:

### 1. Course Information

- a. General information about a course, including the title (with options for displaying differently in the catalog or transcripts), instructor(s) for the course, the course description and the rationale for proposing a new course or modifying an existing one

### 2. Governance

- a. The organizations responsible for oversight and administration of the course and the campus locations that will offer the course. The selected value for the Curriculum Oversight field will determine what organizations need to be involved in the review and approval of the course. Based on this information, KS Curriculum Management determines the approval path for the course creation or modification proposal.

- 3. Course Logistics**
  - a. Specifics such as in what term(s) the course is typically offered, the duration of the course, how grades and assessments will be conducted and recorded for the course, whether a student can audit the course, the course format (lecture, lab, etc.), and the outcome of the course such as credits awarded.
- 4. Learning Objectives**
  - a. Identifies the expected outcomes the student will achieve upon successful completion of the course. Categories can be assigned to learning objectives in order to reflect institutional taxonomies or classification schemes.
- 5. Course Requisites**
  - a. Course requisite rules identify requirements for enrolling in the course, such as specific courses, previous credits, enrollment in a program, grade-point averages, etc. Co-requisites, Recommended Preparation, Antirequisites, Courses that Restrict Credit, and Courses Repeatable for Credit also can be defined.
- 6. Active Dates**
  - a. Specifies the term during which the course will become active for students to enroll, whether the course is a one-time/pilot course, and, if applicable, the term for which the course will no longer be offered.
- 7. Financials**
  - a. Identifies any fees associated with the course as well as the organizations associated with revenue generated and expenditures incurred by offering the course.
- 8. Authors and Collaborators**
  - a. Identifies specific people not on the standard approval path who can edit, comment or view the course proposal.
- 9. Supporting Documents**
  - a. Allows additional files, such as a syllabus, to be attached to the course.

## (CM 2.0) 2.1.1 Course Requisite Rules

Course requisites are structured rules that define a student's eligibility to enroll in a course. What requisites are available for selecting can be configured to suit an institution's specific needs. Examples of requisites defined in KS Curriculum Management as delivered include:

- Courses the student must have completed before enrolling in this course.
- Minimum GPA or specific grades in another course or set of courses.
- Enrollment in a specific program.
- Previously completed courses that would make a student ineligible to take this course.

The rules are defined using drop-down lists to construct structured rules statements. These rules statements can then be used by an online enrollment application to determine if a student is eligible for a course.

## (CM 2.0) 2.2 Course Processes

There are a number of activities that a course navigates throughout its lifetime. KS Curriculum Management 2.0 provides the following processes by which to manage courses:

- 1. Create a Course**
  - a. Someone at the institution proposes a new course, supplying the necessary information, such as requisites, instructors, times offered, financial information, etc. The person who initiates the proposal can identify collaborators who can also edit or review the course information. Once submitted, KS Curriculum Management routes the proposal to the appropriate people based on their roles within the institution.
  - b. Proposals can be created from scratch or can be copied from existing approved or proposed courses. Copied proposals include pre-populated data from the source course. The fields that get copied are configurable by an implementing institution.
  - c. Curriculum managers can be authorized to access special administrative screens to create a course. Administrative screens are streamlined for faster data entry, allow the user to hide non-required fields, and do not require proposed courses to go through the standard review and approval process.
- 2. Modify a Course**
  - a. An approved course may be modified through the proposal process similar to a new course.
  - b. Curriculum managers can also be authorized to access special administrative screens to modify a course. Using these special screens, the curriculum manager can choose whether to modify an existing version of the course or to create a new version, and the modification is not required to go through the standard review and approval process.
- 3. Retire a Course**
  - a. An approved course may be retired through the proposal process, indicating important information like the last term in which the course can be offered and retirement rationale.
  - b. Curriculum managers can be authorized to access a special administrative screen to retire a course. Using this streamlined screen, the curriculum manager can retire a course and is not required to go through the standard review and approval process.

## (CM 2.0) 3. Program Management

KS Curriculum Management provides functionality for creating, modifying, approving, and activating undergraduate and graduate degree-granting academic programs. What information is specified for a program is fully configurable by the institution.

## (CM 2.0) 3.1 Program Information

KS Curriculum Management provides the means to manage the details of a degree program. You can configure KS Curriculum Management to present and require data fields based on your institution's specific business requirements. The following is a summary of the information that is included in KS Curriculum Management as it is delivered:

- 1. Key Program Information**
  - a. Information about a program such as the title, degree type, the title of the program (with alternatives for transcripts and diploma), the term during which it will first be offered and campus locations where it is offered.
- 2. Program Managing Bodies**
  - a. What organizations are responsible for oversight and administration of the program. For modify program proposals, the specified value for the Curriculum Oversight Unit field will determine what organizations need to be involved in the review and approval of the program. Based on this information, KS Curriculum Management determines the approval path for the program modification proposal.
- 3. Specializations**
  - a. Sub-programs within a given program that can be either optional or required to complete the program. All of the information collected for a parent program can be collected for a child specialization. The system verifies that certain specialization information, such as managing bodies and start/end terms, is valid when compared to the parent program.
- 4. Description and Catalog Information**
  - a. A description of the program, including alternative text for publication in the catalog, the duration of the program (four year, two year, etc.) and the core faculty members for the program.
- 5. Program Requirements**
  - a. Requirements for entering the program, progress requirements for satisfactory performance to remain in the program, and requirements for completing the program.
- 6. Learning Objectives**
  - a. Identifies the expected outcomes the student will experience upon successful completion of the program. Learning Objectives can be specific to the program or they can be standard learning objectives that are assigned to more than one program.
- 7. Supporting Documents**
  - a. Allows additional files, such as a syllabus or letter, to be attached to the course.

## (CM 2.0) 3.2 Program Processes

There are a number of activities that a program navigates throughout its lifetime. KS Curriculum Management 2.0 provides the following processes by which to manage programs:

- 1. Create a Program**
  - a. A curriculum manager can use special administrative screens to create a new program, supplying the necessary information. The program can then be approved and activated by the curriculum manager.
  - b. Note that the program creation process does not utilize a proposal process that utilizes review and approval workflow like the course creation process.
- 2. Modify a Program**
  - a. Someone at the institution proposes changes to a program, supplying the necessary information. The person who initiates the proposal can identify collaborators who can also edit or review the course information. Once submitted, KS Curriculum Management routes the proposal to the appropriate people based on their roles within the institution.
  - b. Curriculum managers can be authorized to access special administrative screens to modify a program. The curriculum manager can choose whether to modify an existing version of the program or to create a new version.

## (CM 2.0) 4. Course Set Management

KS Curriculum Management allows you to create groups of courses that can then be referenced by other processes. In Release 2.0, course sets are used when creating course requisite rules or program requirements. The following are example uses for this feature:

- 1. Course Requisite Rules: Multiple Choice Prerequisites**
  - a. A set of courses from which a student must take one, but not all, to be eligible to enroll in a subsequent course.
- 2. Program Requirements: Core Courses**
  - a. A defined set of courses that all students, or all students within a program, must complete.

Course sets may be fixed or dynamic lists of courses. A fixed course set is defined by explicitly adding courses to the list by course number (MATH101, MATH102, MATH103, etc.). A dynamic course set is defined by applying some criteria, such as any 100-level course in the Mathematics department. As courses are added that meet the criteria, they are automatically added to the dynamic course set.

## (CM 2.0) 5. Learning Objectives Management

Learning objectives provide a way to manage the inventory of learning objectives. A learning objective is a statement that describes the knowledge, skills or abilities that a student would expect to gain by participating in a course or program. KS Curriculum Management provides tools to categorize, search, and re-use learning objectives.

The key principles of a learning objective include:

1. It is specific.
2. It is measurable/observable
3. It is attainable
4. It is relevant and results-oriented
5. It is targeted to the learner and the desired level of learning.

Learning objectives can range from a simple statement describing the student's expected mastery of some aspect of the subject to an specific certification. A learning objectives can be attached to any number of programs or courses that have that objective as an expected outcome.

### Learning Objective Hierarchies

1. Learning objectives can be maintained in hierarchies with levels and sublevels if desired. Learning objectives have various levels of abstraction and specificity and systems have evolved for describing the hierarchies of learning objectives (e.g., Fink's Taxonomy and Bloom's Taxonomy). An institution can decide whether they are interested in using taxonomies to manage the relationships between learning objectives.

#### **Learning Objectives Categories**

1. Since many learning objectives are short, free-form statements, categories provide a means for authors to provide more information about their learning objective such as accreditation categories, skill-based categories, subject-matter categories. These categories facilitate discovery of learning objectives but also enhanced reporting capabilities for the institution.

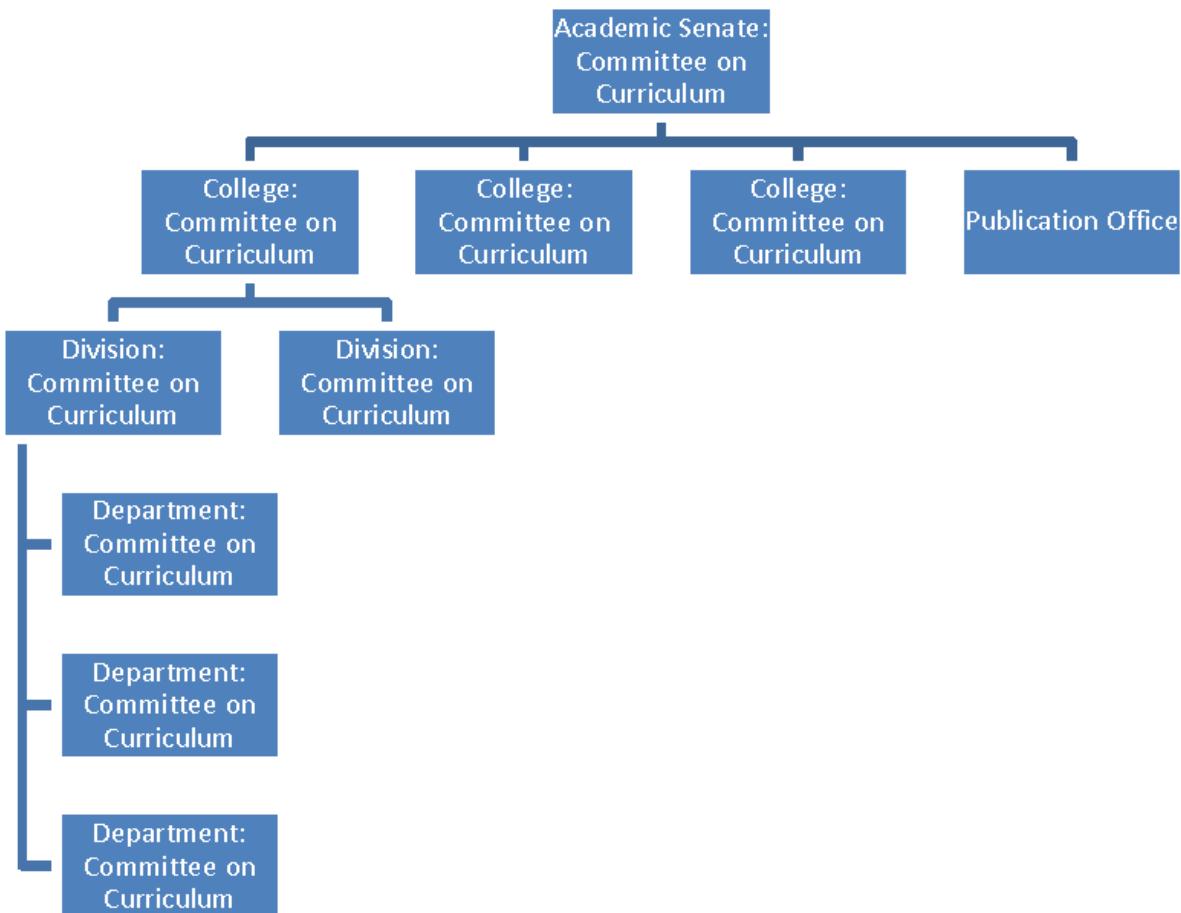
## **(CM 2.0) 6. Organization, People and Roles Management**

The organizational structure of the institution and the specific people and their roles within that organization, are the keys to defining the workflow processes through which course and program changes are routed. KS Curriculum Management uses Kuali Identity Management (KIM) and Kuali Enterprise Workflow (KEW) to determine who can propose, review, modify and approve courses and programs. These responsibilities are not tied directly to individuals, but rather to roles that individuals occupy in the organization.

For example, if a new course is proposed for the Biological Sciences department, KS Curriculum Management determines the approval path for that course based on what it knows about the organizational structure. The first step may require review and approval by the Chair of the Biological Sciences department's Committee on Curriculum. Once approved, the proposal moves on to the institution's Committee on Curriculum. KS Curriculum Management determines the current members and chair of the committee and presents the proposal to them for review, comment or approval. After the appropriate approvals have been obtained, KS Curriculum Management determines the next step is to present the proposal to the Publication Office for final approval.

As peoples' roles change, KS Curriculum Management will be able to react to these changes and route the proposal to the appropriate people. For example, if a member of the Committee on Curriculum is replaced with a new member, KS Curriculum Management will present proposals to the new member instead. By keeping the organizational structure, people and roles current, an institution can ensure that course proposals are routed to the appropriate people.

You can configure KIM and KEW to match your institution's specific needs. Without modification, KS Curriculum Management assumes the following organizational structure and roles for managing workflow.



## (CM 2.0) 7. Reference Implementation Configuration

- (CM 2.0) Field Requiredness by Workflow Node Configuration
- (CM 2.0) Organization Configuration
- (CM 2.0) Subject Areas and Curriculum Org Relationship Configuration
- (CM 2.0) User Configuration
- (CM 2.0) Workflow Configuration and Associated Users

### (CM 2.0) Field Requiredness by Workflow Node Configuration

#### Create and Modify Course (Non-Admin) Proposal Fields and Requiredness

UI Section	UI Field	Required for Proposal State	Required for Workflow Node
Course Info	Proposal Title	Saved	n/a
Course Info	Course Title	Saved	n/a
Course Info	Transcript Course Title	Enroute	College
Course Info	Subject Code	Enroute	Proposer
Course Info	Course Number	Enroute	Division
Course Info	Cross Listed		
Course Info	Jointly Offered		

Course Info	Version Codes		
Course Info	Instructor(s)		
Course Info	Description	Enroute	Proposer
Course Info	Proposal Rationale	Enroute	Proposer
Governance	Campus Locations	Enroute	Senate
Governance	Curriculum Oversight	Enroute	Proposer
Governance	Admin Org		
Course Logistics	Term		
Course Logistics	Duration Type		
Course Logistics	Duration Count		
Course Logistics	Assessment Scale	Enroute	Proposer
Course Logistics	Audit		
Course Logistics	Pass Fail Transcript Grade		
Course Logistics	Final Exam Status	Enroute	Proposer
Course Logistics	Final Exam Rationale	Only if Final Exam Status is Alternate or No Final Exam	Whoever picks Alternate of No Final Exam for Final Exam Status
Course Logistics	Outcome :: Type	Enroute	Proposer
Course Logistics	Outcome :: Value(s)	Enroute	Proposer
Course Logistics	Format :: Activity Type	Enroute	Department
Course Logistics	Format :: Contact Hours		
Course Logistics	Format :: Frequency		
Course Logistics	Format :: Duration Type		
Course Logistics	Format :: Duration Count		
Course Logistics	Format :: Anticipated Class Size		
Learning Objectives	LO Description		
Learning Objectives	LO Category		
Course Requisites	Student Eligibility + Prerequisite		
Course Requisites	Corequisite		
Course Requisites	Recommended Preparation		
Course Requisites	Antirequisite		
Course Requisites	Courses that Restrict Credit		
Course Requisites	Repeatable for Credit		
Active Dates	Start Term	Enroute	Proposer
Active Dates	Pilot Course		
Active Dates	End Term	Only if Pilot Course is checked	Whoever checks Pilot Course
Active Dates	End Term for Old Version (when modifying a course)	Enroute	Publication Office
Financials	Justification of Fees	Only with Fees present	Whoever adds Fees
Financials	Fees: Fee Type		

Financials	Fees: Rate Type		
Financials	Fees: Amount		
Financials	Revenue :: Source		
Financials	Revenue :: Percentage		
Financials	Expenditures :: Expenditure Org		
Financials	Expenditures :: Percentage		
Authors & Collaborators	n/a		
Supporting Documents	n/a		

## Modify Program Proposal Fields and Requiredness

UI Section	UI Field	MAJOR Required for Proposal State	MAJOR Required for Workflow Node	SPECIALIZATION Required for Proposal State	SPECIALIZATION Required for Workflow Node
Proposal Information	Proposal Title	Saved	n/a	n/a	n/a
	Type of Modification	Enroute	Proposer	n/a	n/a
	Proposal Abstract			n/a	n/a
	Proposal Rationale	Enroute	Proposer	n/a	n/a
Change Impact	Related Course Changes			n/a	n/a
	Impacted Units			n/a	n/a
	Impacted Articulation or Transfer Programs			n/a	n/a
	Student Transition Plans			n/a	n/a
Key Program Info	Institution				
	Credential Program				
	Level				
	Code	Enroute	Publication Office	Enroute	Publication Office
	Classification	Saved	n/a	Saved	n/a
	Degree Type	Enroute	Proposer	Enroute	Proposer
	Title Full	Saved	n/a	Saved	n/a
	Title Short	Enroute	Publication Office	Enroute	Publication Office
	Title Transcript	Enroute	Publication Office	Enroute	Publication Office
	Title Diploma	Enroute	Publication Office	Enroute	Publication Office
	Start Term	Enroute	Proposer	Enroute	Proposer
	End Inst Admit Term				
	End Program Entry Term				
	End Program Enroll Term				
	Location	Enroute	Proposer	Enroute	Proposer

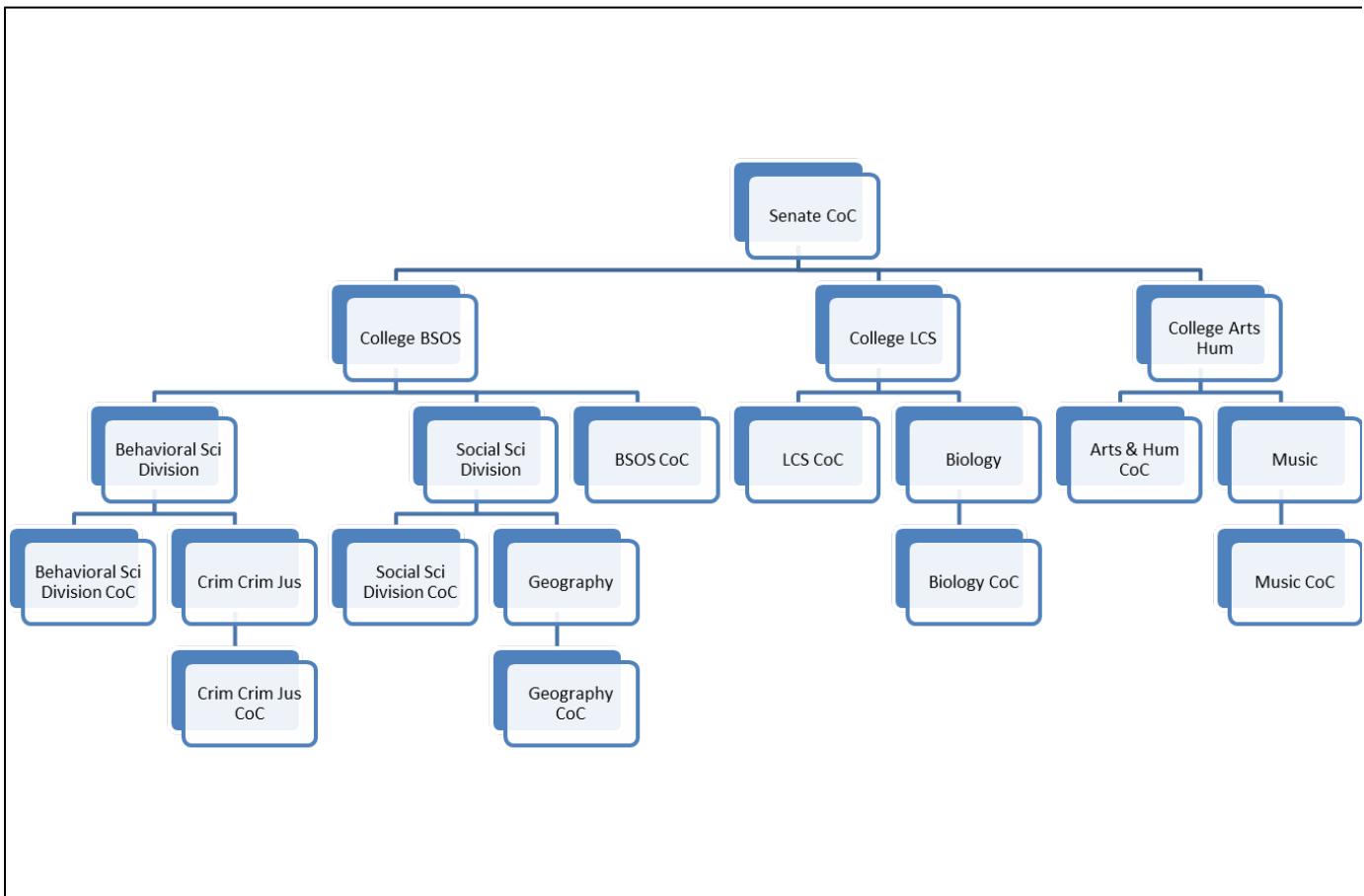
	CIP 2000				
	CIP 2010	Enroute	System Office	Enroute	System Office
	HEGIS	Enroute	System Office	Enroute	System Office
Program Managing Bodies	Curriculum Oversight Unit	Enroute	Proposer	Enroute	Proposer
	Curriculum Oversight Division	Enroute	Proposer	Enroute	Proposer
	Student Oversight Unit	Enroute	Department	Enroute	Department
	Student Oversight Division	Enroute	Department	Enroute	Department
	Deployment Unit	Enroute	Department	Enroute	Department
	Deployment Division	Enroute	Department	Enroute	Department
	Financial Resources Unit	Enroute	College	Enroute	College
	Financial Resources Division	Enroute	College	Enroute	College
	Financial Control Unit	Enroute	College	Enroute	College
	Financial Control Division	Enroute	College	Enroute	College
Specializations	Completion required?				
Description and Catalog Information	Program Description	Enroute	Proposer		
	Catalog Description				
	Core Faculty Members				
	Publication Targets	Enroute	Publication Office	Enroute	Publication Office
	Full Time/Part Time				
	Duration				
	Duration Count				
	Duration Notes				
	More info				
Program Requirements	Entrance Requirements				
	Satisfactory Progress Requirements				
	Completion Requirements				
Learning Objectives	LO text	Enroute	Proposer		
	LO category				
Supporting Documents	File Name				

	Description				
--	-------------	--	--	--	--

## (CM 2.0) Organization Configuration

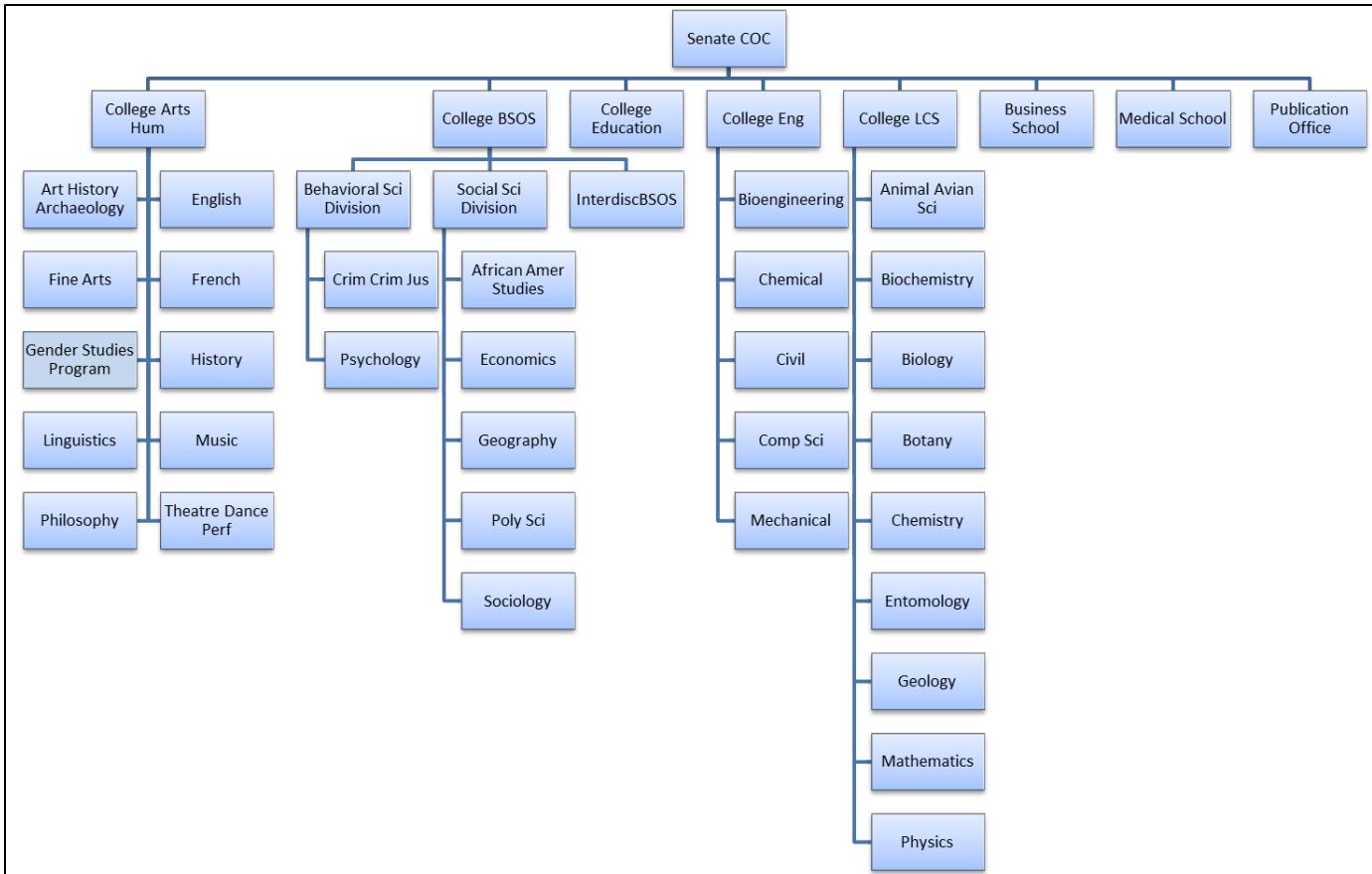
### Curriculum Hierarchy for Organizations Configured for Workflow

Only certain organizations have been fully configured with all of the users and organizational relationships required to support workflow routing. The curriculum hierarchy uses "curriculum parent" and "curriculum child" relationship types. The following chart depicts the relationships among the configured organizations.



### Full Curriculum Hierarchy

The following chart depicts the curriculum hierarchy from the Senate Committee on Curriculum (CoC) down through the colleges, divisions (where applicable) and departments. Every college, division, and department has a curriculum committee; however, these committees are not depicted on the chart.



## (CM 2.0) Subject Areas and Curriculum Org Relationship Configuration

Subject Area Code	Subject Area Name	Org Relation Type (1)	Organization(s)
AASP	African American Studies	All	African American Studies Dept
AAST	Asian American Studies	All	College of Behavioral and Social Sciences
ACCT	Accounting	All	Business School
ANSC	Animal Science	All	Animal and Avian Science Dept
ARHU	Arts and Humanities	All	College of Arts and Humanities
ARTS	Fine Arts	All	Fine Arts Dept
BCHM	Biochemistry	All	Biochemistry Dept
BEES	Behavior, Ecology, Evolution and Systematics	All	Biology Dept
BENG	Biomedical Engineering	All	College of Life and Chemical Sciences College of Engineering
BIOE	Bioengineering	All	Bioengineering Dept
BIOM	Biometrics	All	Biology Dept
BMGT	Business Management	All	Business School
BOTA	Botany	All	Botany Dept
BSCI	Biological Sciences	All	Biology Dept
CBMG	Cell Biology and Molecular Genetics	All	Biology Dept

CCJS	Criminal Justice	All	Criminology and Criminal Justice Dept
CHEE	Chemical Engineering	All	Chemical Engineering Dept
CHEM	Chemistry	All	Chemistry Dept
CIVI	Civil Engineering	All	Civil Engineering Dept
CSCI	CompSci	All	Computer Science Dept
ECON	Economics	All	Economics Dept
EDUC	Education	All	College of Education
ENGL	English	All	English Dept
ENTM	Entomology	All	Entomology Dept
FINA	Finance	All	Finance Dept
FREN	French	All	French Dept
GEMS	Gemstone	All	College of Behavioral and Social Science Office of Undergraduate Studies
GEOG	Geography	All	Geography Dept
GEOL	Geology	All	Geology Dept
GVPT	Government	All	Political Science Dept
HIST	History	All	History Dept
HONR	Honors	One	College of Arts and Humanities Office of Undergraduate Studies
INTB	International Business	All	Business School
LING	Linguistics	All	Linguistics Dept
MARK	Marketing	All	Business School
MATH	Math	All	Mathematics Dept
MEES	Marine-Estuarine-Environmental Sciences	All	Biology Dept
MENG	Mechanical Engineering	All	Mechanical Engineering Dept
MOCB	Molecular and Cellular Biology	All	Biology Dept
MUED	Music Education	All	Music Dept
MUET	Ethnomusicology	All	Music Dept
MUSC	Music	All	Music Dept
MUSP	Music Performance	All	Music Dept
NACS	Neuroscience and Cognitive Science	One	Biology Dept Psychology Dept
PHIL	Philosophy	All	Philosophy Dept
PHYS	Physics	All	Physics Dept
POLI	Political Science	All	Political Science Dept
PSYC	Psychology	All	Psychology Dept
PUAD	Public Administration	All	Political Science Dept
ROBT	Robotics	All	Mechanical Engineering Dept

SOCY	Sociology	All	Sociology Dept
SOWK	Social Work	All	Sociology Dept
STAT	Statistics	All	Mathematics Dept
THET	Theatre	All	Theatre, Dance, and Performance Arts Dept
UNIV	University Studies	One	Undergraduate Studies College of Arts and Humanities College of Behavioral and Social Science

**Notes:**

(1) Org Relation Type: When a particular subject code is used in a course proposal, the application will enforce that either **one** or **all** of the organizations listed must be included as values in the Curriculum Org field.

## (CM 2.0) User Configuration

### User Category Descriptions

- Generic user
  - Can generate create and modify course proposals
  - Can generate modify program proposals
  - Can search for course and program proposals
  - Can search, browse, and view approved courses and programs, as well as course/program version history
  - Can copy existing courses and course proposals to a new course proposal
  - Can manage course sets and learning objective categories
  - Can generate dependency analyses
- Faculty/staff user
  - Can perform generic user functions indicated above
  - Can view course and program proposals that have been cancelled, withdrawn, rejected, or approved
- Admin user
  - Can perform generic and faculty/staff user functions indicated above
  - Can use administrator screens to create, modify, and retire courses
  - Can use administrator screens to create and modify programs
  - Can modify existing versions of courses and programs
- Workflow user :: In addition to belonging to one of the three previous categories, a user can be a workflow user.
  - Can review or approve course and program proposals for his/her organization, as detailed on [Workflow Configuration and Associated Users](#)
  - Can view course and program proposals that are submitted to workflow, if the user is part of the proposal's workflow

### Users

Username	User Category	Workflow User?
admin	Faculty/Staff	No
admin1	Generic	No
admin2	Generic	No
dev1	Generic	Yes
dev2	Generic	Yes
director	Generic	No
doug	Generic	No
earl	Generic	No
edna	Generic	No
employee	Generic	No
eric	Generic	No
erin	Generic	No

fran	Faculty/Staff	No
frank	Faculty/Staff	No
fred	Faculty/Staff	No
idm1	Generic	No
idm2	Generic	No
idm 3	Generic	No
1	Generic	No
5	Generic	No
kuluser	Generic	No
newaccountuser	Generic	No
notsys	Generic	No
notsysadm	Generic	No
quickstart	Generic	No
supervisor	Generic	No
test1	Generic	Yes
test2	Generic	Yes
testadmin1	Generic	No
testadmin2	Generic	No
testuser1	Generic	Yes
testuser10	Faculty	Yes
testuser2	Generic	Yes
testuser3	Generic	Yes
testuser4	Generic	Yes
testuser5	Generic	Yes
testuser6	Generic	Yes
testuser7	Faculty/Staff	Yes
testuser8	Faculty/Staff	Yes
testuser9	Faculty/Staff	Yes
user1	Generic	Yes
user2	Generic	Yes
user3	Generic	Yes
user4	Generic	Yes
user5	Faculty/Staff	Yes
user6	Faculty/Staff	Yes
user7	Faculty/Staff	Yes
user8	Faculty/Staff	Yes

## (CM 2.0) Workflow Configuration and Associated Users

- Create Course Workflow
- Modify Course Workflow

- Modify Program Workflow

## Create Course Workflow

Dept Name	Subject Area(s)	Department Review	Division Review	College Review	Senate Review	Publication Review
Biology	BEES BSCI CBMG MEES MOCB NACS	<b>Biology Dept COC</b> Chair: User One (user1) Member: User Two (user2)	N/A	<b>College of Life &amp; Chemical Sciences COC</b> Chair: User Three (user3) Member: user4 user4 (user4)	<b>Senate COC</b> Chair: Tester One (test1) Member: Tester Two (test2)	<b>Publication Office</b> Chair: Developer One (dev1) Member: Developer Two (dev2)
Criminology & Criminal Justice	CCJS	<b>Criminology &amp; Criminal Justice Dept COC</b> Chair: Test User 7 (testuser7) Member: Test User 8 (testuser8)	<b>Behavioral Science Division COC</b> Chair: Test User 9 (testuser9) Member: Test User 10 (testuser10)	<b>College of Behavioral &amp; Social Science COC</b> Chair: Test User 5 (testuser5) Member: Test User 6 (testuser6)	<b>Senate COC</b> Chair: Tester One (test1) Member: Tester Two (test2)	<b>Publication Office</b> Chair: Developer One (dev1) Member: Developer Two (dev2)
Geography	GEOG	<b>Geography Dept COC</b> Chair: Test User 1 (testuser1) Member: Test User 2 (testuser2)	<b>Social Science Division COC</b> Chair: Test User 3 (testuser3) Member: Test User 4 (testuser4)	<b>College of Behavioral &amp; Social Science COC</b> Chair: Test User 5 (testuser5) Member: Test User 6 (testuser6)	<b>Senate COC</b> Chair: Tester One (test1) Member: Tester Two (test2)	<b>Publication Office</b> Chair: Developer One (dev1) Member: Developer Two (dev2)
Music	MUED MUET MUSC MUSP	<b>Music Dept COC</b> Chair: User Five (user5) Member: User Six (user6)	N/A	<b>College of Arts &amp; Humanities COC</b> Chair: User Seven (user7) Member: User Eight (user8)	<b>Senate COC</b> Chair: Tester One (test1) Member: Tester Two (test2)	<b>Publication Office</b> Chair: Developer One (dev1) Member: Developer Two (dev2)

## Modify Course Workflow

Dept Name	Subject Area(s)	Document Organization Review	Publication Decision Review	Department Review	Division Review	College Review	Senate Review	Publication Review
Biology	BEES BSCI CBMG MEES MOCB NACS	<b>Biology Dept COC</b> Chair: User One (user1)  Member: User Two (user2)	<b>Publication Office</b> Chair: Developer One (dev1) Member: Developer Two (dev2)	<b>Biology Dept COC</b> Chair: User One (user1) Member: User Two (user2)	N/A	<b>College of Life &amp; Chemical Sciences COC</b> Chair: User Three (user3) Member: user4 user4 (user4)	<b>Senate COC</b> Chair: Tester One (test1) Member: Tester Two (test2)	<b>Publication Office</b> Chair: Developer One (dev1) Member: Developer Two (dev2)

Criminology & Criminal Justice	CCJS	<b>Criminology &amp; Criminal Justice Dept COC</b> Chair: Test User 7 (testuser7) Member: Test User 8 (testuser8)	<b>Publication Office</b> Chair: Developer One (dev1) Member: Developer Two (dev2)	<b>Criminology &amp; Criminal Justice Dept COC</b> Chair: Test User 7 (testuser7) Member: Test User 8 (testuser8)	<b>Behavioral Science Division COC</b> Chair: Test User 9 (testuser9) Member: Test User 10 (testuser10)	<b>College of Behavioral &amp; Social Science COC</b> Chair: Test User 5 (testuser5) Member: Test User 6 (testuser6)	<b>Senate COC</b> Chair: Tester One (test1) Member: Tester Two (test2)	<b>Publication Office</b> Chair: Developer One (dev1) Member: Developer Two (dev2)
Geography	GEOG	<b>Geography Dept COC</b> Chair: Test User 1 (testuser1) Member: Test User 2 (testuser2)	<b>Publication Office</b> Chair: Developer One (dev1) Member: Developer Two (dev2)	<b>Geography Dept COC</b> Chair: Test User 1 (testuser1) Member: Test User 2 (testuser2)	<b>Social Science Division COC</b> Chair: Test User 3 (testuser3) Member: Test User 4 (testuser4)	<b>College of Behavioral &amp; Social Science COC</b> Chair: Test User 5 (testuser5) Member: Test User 6 (testuser6)	<b>Senate COC</b> Chair: Tester One (test1) Member: Tester Two (test2)	<b>Publication Office</b> Chair: Developer One (dev1) Member: Developer Two (dev2)
Music	MUED MUET MUSC MUSP	<b>Music Dept COC</b> Chair: User Five (user5)  Member: User Six (user6)	<b>Publication Office</b> Chair: Developer One (dev1) Member: Developer Two (dev2)	<b>Music Dept COC</b> Chair: User Five (user5) Member: User Six (user6)	N/A	<b>College of Arts &amp; Humanities COC</b> Chair: User Seven (user7) Member: User Eight (user8)	<b>Senate COC</b> Chair: Tester One (test1) Member: Tester Two (test2)	<b>Publication Office</b> Chair: Developer One (dev1) Member: Developer Two (dev2)

## Modify Program Workflow

Dept Name	Reference Majors	Department Review	Division Review	College Review	Senate Review	President's Office Review	System Office Review	Publication Review
Biology Dept	BISI BSCI	<b>Biology Dept COC</b> Chair: User One (user1) Member: User Two (user2)	N/A	<b>College of Life &amp; Chemical Sciences COC</b> Chair: User Three (user3) Member: user4 user4 (user4)	<b>1. Senate Curriculum Manager</b> Curriculum Admin: Developer Two (dev2) <b>2. Senate COC</b> Chair: Tester One (test1) Member: Tester Two (test2)	<b>Senate Curriculum Manager</b> Curriculum Admin: Developer Two (dev2)	<b>Senate Curriculum Manager</b> Curriculum Admin: Developer Two (dev2)	<b>Publication Office</b> Chair: Developer One (dev1) Member: Developer Two (dev2)

Criminology & Criminal Justice Dept	CCJS CRJM	Criminology & Criminal Justice Dept COC Chair: Test User 7 (testuser7) Member: Test User 8 (testuser8)	Behavioral Science Division COC Chair: Test User 9 (testuser9) Member: Test User 10 (testuser10)	College of Behavioral & Social Science COC Chair: Test User 5 (testuser5) Member: Test User 6 (testuser6)	1. Senate Curriculum Manager Curriculum Admin: Developer Two (dev2)  2. Senate COC Chair: Tester One (test1) Member: Tester Two (test2)	Senate Curriculum Manager Curriculum Admin: Developer Two (dev2)	Senate Curriculum Manager Curriculum Admin: Developer Two (dev2)	Publication Office Chair: Developer One (dev1) Member: Developer Two (dev2)
Geography Dept	GEOG GRPH	Geography Dept COC Chair: Test User 1 (testuser1) Member: Test User 2 (testuser2)	Social Science Division COC Chair: Test User 3 (testuser3) Member: Test User 4 (testuser4)	College of Behavioral & Social Science COC Chair: Test User 5 (testuser5) Member: Test User 6 (testuser6)	1. Senate Curriculum Manager Curriculum Admin: Developer Two (dev2)  2. Senate COC Chair: Tester One (test1) Member: Tester Two (test2)	Senate Curriculum Manager Curriculum Admin: Developer Two (dev2)	Senate Curriculum Manager Curriculum Admin: Developer Two (dev2)	Publication Office Chair: Developer One (dev1) Member: Developer Two (dev2)
Music Dept	MUSP	Music Dept COC Chair: User Five (user5) Member: User Six (user6)	N/A	College of Arts & Humanities COC Chair: User Seven (user7) Member: User Eight (user8)	1. Senate Curriculum Manager Curriculum Admin: Developer Two (dev2)  2. Senate COC Chair: Tester One (test1) Member: Tester Two (test2)	Senate Curriculum Manager Curriculum Admin: Developer Two (dev2)	Senate Curriculum Manager Curriculum Admin: Developer Two (dev2)	Publication Office Chair: Developer One (dev1) Member: Developer Two (dev2)

## (CM 2.0) 8. Course and Proposal States

This document clarifies Course States, Proposal States and Workflow Status, and their relationships and definitions.

**WORKFLOW STATES:** Workflow Status is defined by KEW and not under the purview of KS to change.

Workflow Status	Description	Proposal States
<b>Saved</b>	document has been created by author prior to routing	Saved
<b>Enroute</b>	document is being routed	Enroute

<b>Processed</b>	document has received all required Approvals	Approved
<b>Final</b>	document has received all required Approvals and has all Acknowledgments cleared; proceed to post-processing	Approved
<b>Disapproved</b>	document has been Disapproved by reviewer; routing stops	Rejected (by Reviewer), Cancelled (by Initiator), Withdrawn (by Initiator)
<b>Exception</b>	issue/problem with routing; document requires intervention by Exception User/Workgroup	Exception

**PROPOSAL STATES:** Proposal states are under the purview of KS and, to the extent possible, are aligned with workflow status.

Proposal States	Description	State set when:	Workflow Status(es)
<b>Saved</b>	the <b>proposal</b> has been saved but not routed	the proposal is first Saved by the initiator	Saved
<b>Cancelled</b>	the <b>proposal</b> has been saved, but the initiator wants to cancel the proposal instead of submitting	the proposal is Cancelled by the initiator	Disapproved (by Initiator)
<b>Enroute</b>	the <b>proposal</b> is being routed	the proposal is Submitted to workflow	Enroute
<b>Withdrawn</b>	the <b>proposal</b> has been withdrawn by initiator prior to final approval	the enroute proposal is Withdrawn by the initiator	Disapproved (by Initiator)
<b>Approved</b>	the <b>proposal</b> has received all required Approvals	the last reviewer the on Route Log with an Approve Action Request approves the proposal	Processed, Final

<b>Rejected</b>	the <b>proposal</b> has been rejected	a reviewer on the Route Log with an Approve Action Request disapproves the proposal	Disapproved
<b>Exception</b>	there is an issue with the routing	routing fails	Exception

#### COURSE STATES:

The following is a summary of changes (and brief rationale) implemented in R1.2. (Leaving this section intact for 2.0 documentation, providing context for institutions that might find it useful.)

- Change business meaning of Approved, Active, and Retired course states.
- Create course state 'Not Approved.' Currently, proposals that are cancelled, rejected, or withdrawn sit in a state of 'Submitted.' This change would help the user make better sense of what happened to such proposals when viewing course version history.
- Deprecate course state 'Submitted.' All courses will be 'Draft' until they become 'Not Approved' or 'Approved.' Determination of field requiredness would become based on proposal states.
- Rename course state 'Inactive' to 'Suspended.' This more accurately describes to the user what happened to the course, and matches program states.
- Create course state grouping 'Latest States.' As implemented in program, 'latest states' reflects the ONE version that is either active, suspended, or retired. We can then limit actions or system behaviors to be based on latest states, i.e. which courses appear in search results.
- Implement the following constraints on course states. These match constraints on program states.
  - Only one DRAFT at a time. This will disallow course modification when a version with courseState = Draft already exists.
  - Only one of the following at a time (Active, Suspended, Retired)
  - OK to have multiple versions that are Superseded

Implementing the above changes then positions us to address the following changes to functionality:

- Change field requiredness for course proposals to be based on proposal states of Saved and Enroute. Make appropriate changes to requiredness indicators in UI.
- Within field requiredness for proposalState = Enroute, introduce requiredness by workflow node. Make appropriate changes to requiredness indicators in UI.

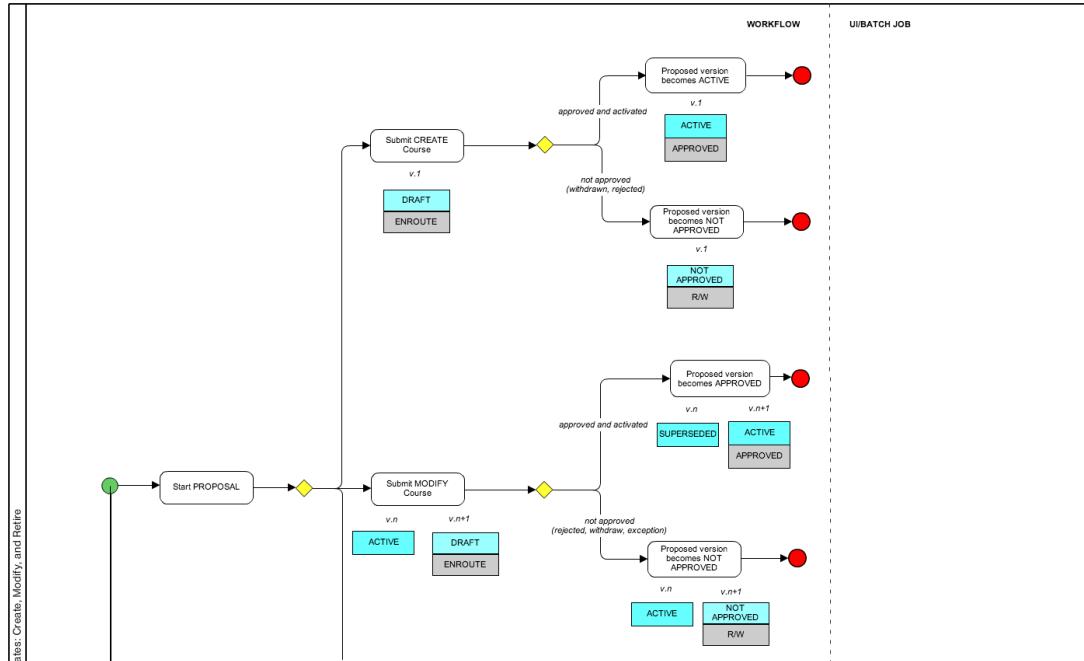
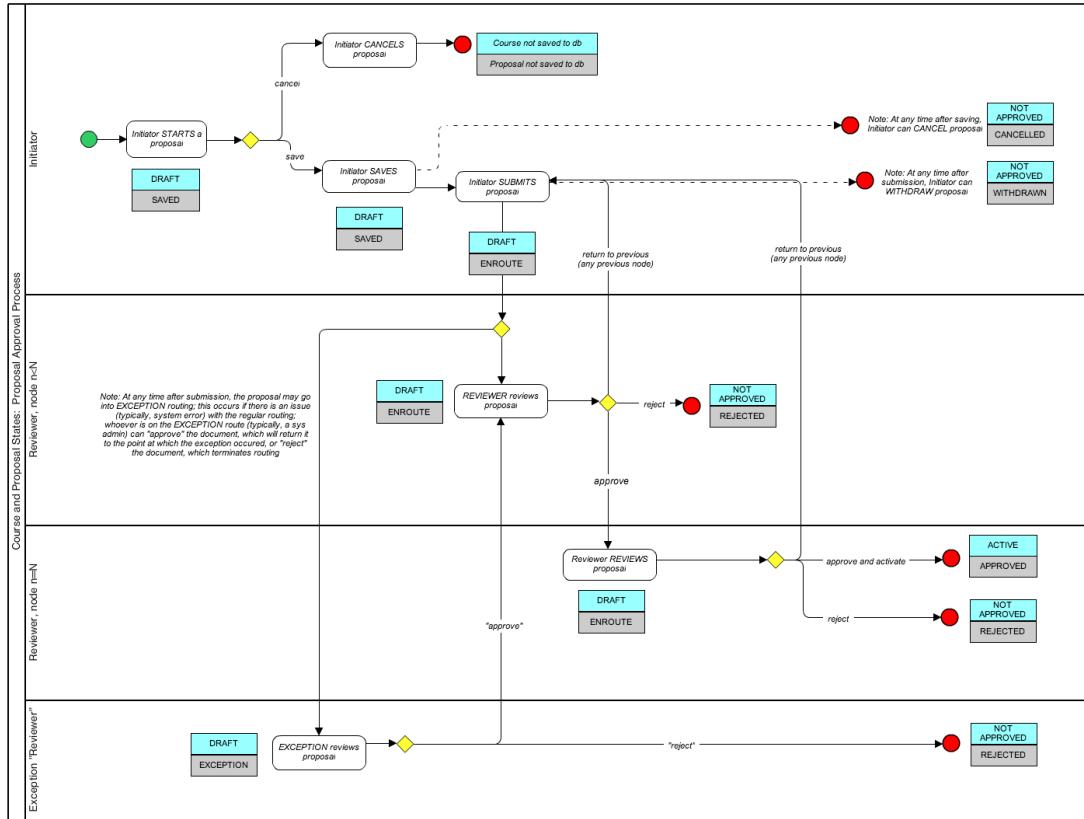
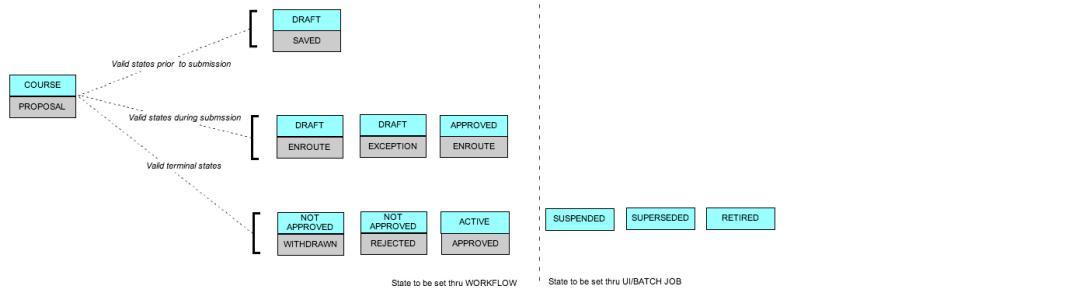
The future and current states and their definitions are given below:

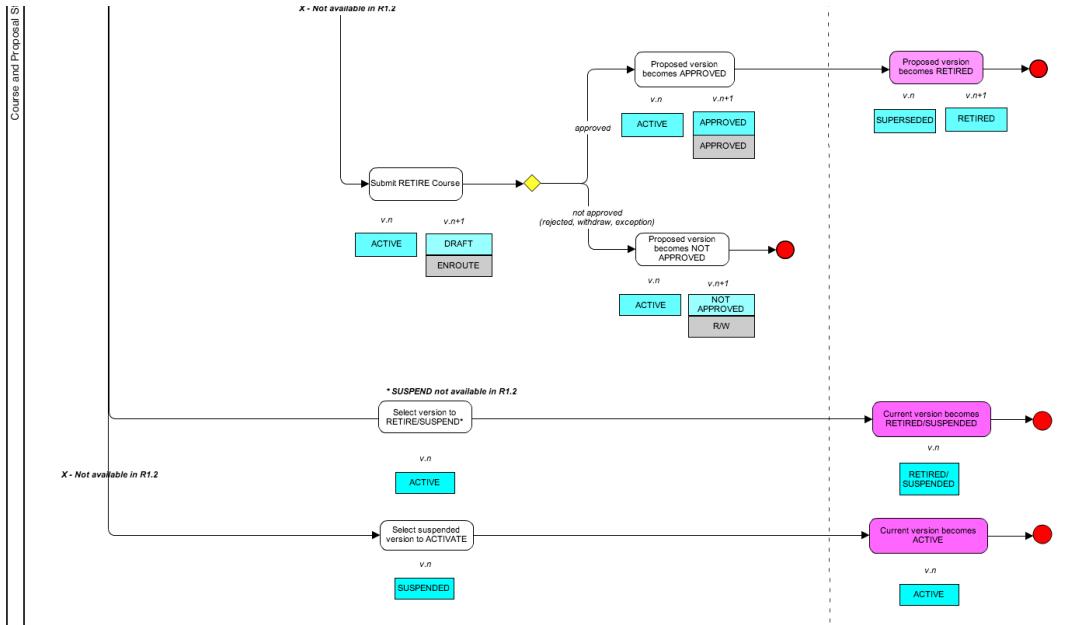
Course States	Description	State change expected when:	Allowed Actions	Valid Next States	Description	State change expected when:	Allowed Actions
Draft	<b>course</b> is in the process of being proposed/created	course proposal is saved OR course is entered into system via screen(s)	<u>Initiator:</u> Cancel Submit Withdraw Copy to New Proposal  <u>Reviewer:</u> Approve Reject Return to Previous Node  <u>Admin:</u> Blanket Approve Modify without Version (to be implemented for admin screens only)	Not Approved Active	<b>course proposal</b> has been started and saved	course proposal is saved	Cancel Submit Withdraw

<b>Not Approved</b>	<b>course creation or modification was not approved</b>	course proposal cancelled or withdrawn by initiator OR rejected by reviewer.	Copy to New Proposal	None	N/A (did not exist)	N/A (did not exist)	N/A (did not exist)
<b>Approved</b>	<b>course has been approved (i.e. completed curriculum review business process)</b>	R1.2 reference implementation will not utilize Approved state. Courses will move directly from Draft to Active. An implementing institution could have a two-step process in which courses are first Approved, then Active.	R1.2 reference implementation will not utilize Approved state	R1.2 reference implementation will not utilize Approved state	<b>course</b> has been approved but is not eligible for offering, typically due to the lag between the approval process and the start term	course proposal has been approved or course is set to approved via screen(s)	Modify (only if latest version) Activate
<b>Active</b>	<b>course has been prepared for offering (i.e. reviewed and updated by a curriculum administrator to ensure course data is consistent with institutional policies and practices)</b>	course proposal has been approved at Publication Office node of reference workflow	Modify with Version Modify without Version (for admin screens only) Suspend Retire Copy to New Proposal	Draft Suspended Superseded Retired	<b>course</b> is eligible for offering	set via batch job when start term is effective OR set manually via screen(s)	Modify Inactive (not yet implemented) Retire (not yet implemented)
<b>Suspended (was Inactive in R1.1)</b>	<b>course has been approved but is temporarily not eligible for offering</b>	Suspended will not be implemented in R1.2	Activate Modify with Version Modify without Version (for admin screens only) Retire Copy to New Proposal	Active Draft Retired	same	set manually via screen(s)	Not yet implemented
<b>Superseded</b>	<b>course has been superseded by newer version</b>	set via batch job when newer version becomes active	Modify without Version (for admin screens only)	None	same	set via batch job when newer version becomes active OR set manually via screen(s)	None
<b>Retired</b>	<b>course has been approved for retirement</b>	retire course administrative action is approved  Retire Course by proposal will not be implemented in R1.2	Modify with Version Modify without Version (for admin screens only) Copy to New Proposal	Draft	<b>course</b> has been retired and is no longer eligible for offering	retire course proposal is approved OR set manually via screen(s)	Not yet implemented

 Latest States	Grouping of the latest states. Cannot have more than one version among these states: active, suspended, & retired. Default search result should include just these versions as well.			N/A (did not exist)	N/A (did not exist)	N/A (did not exist)
-------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--	--	---------------------	---------------------	---------------------

The figure below show the when the various course and proposal states are valid, and when they are set through the proposal process. **Figure has been updated to reflect R1.2 changes and has not changed for 2.0. See version 18 for what was implemented in R1.1.**





## R1.2 Allowable Actions and Next States

Current Course State	Action	Next State of this Course Version for this Action
Draft	Cancel (before save)	N/A
	Cancel (after save)	Not Approved
	Submit	Draft (no state change)
	Withdraw	Not Approved
	Approve (at last node)	Active
	Reject (at any node)	Not Approved
	Return to Previous Node	Draft (no state change)
	Blanket Approve	Approved
	Modify without Version	Draft (no state change)
	Copy to New Proposal	Draft (no state change)
Not Approved	Copy to New Proposal	Not Approved (no state change)
Approved	None; Approved will not be utilized as a state in R1.2 reference implementation	Active
Active	Modify with Version	New version: Draft Old version: Superseded, upon activation of subsequent version
	Modify without Version	Active (no state change)
	Suspend	Suspended
	Retire	Retired
	Copy to New Proposal	Active (no state change to "source" course)
Suspended	Activate	Active

	Modify with Version	New version: Draft Old version: Superseded, upon activation of subsequent version
	Modify without Version	Suspended (no state change)
	Retire	Retired
	Copy to New Proposal	Suspended (no state change)
Retired	Modify with Version	New version: Draft Old version: Superseded, upon activation of subsequent version
	Modify without Version	Retired (no state change)
	Copy to New Proposal	Retired (no state change)

**From Norm's Types and States used by Courses page**

name	desc	include	xmlObject	effective	expiration	status	comments
draft	Draft either public or private	TRUE	clInfo	1/1/2010		Released	not sure if private/public applies to program?
draft.private	Exploratory/Private scratch pad	FALSE	clInfo	1/1/2010		Revisit	Transitioned to proposal state?
draft.public	Entered as draft but not yet submitted	FALSE	clInfo	1/1/2010		Revisit	Transitioned to proposal state?
submitted	Submitted but not yet approved	FALSE	clInfo	1/1/2010		Revisit	Transitioned to proposal state?
withdrawn	Withdrawn (anytime before activated)	FALSE	clInfo	1/1/2010		Revisit	Transitioned to proposal state?
approved	Approved	FALSE	clInfo	1/1/2010		Revisit	Transitioned to proposal state?
rejected	Not approved (rejected)	FALSE	clInfo	1/1/2010		Revisit	Do we need here some indication of why it was not approved? i.e. disapproved vs. not acted on because submitted too late, etc.
active	Active ready to be used and published	TRUE	clInfo	1/1/2010		Released	

superseded	Old version that has been replaced by a newer version. Students may be grandfathered into this program but new students must enroll in the new program	TRUE	clInfo	1/1/2010		Released	When a new version of a program is approved the existing one needs to be marked as superseded.
latest states	Grouping of the latest states. Cannot have more than one version among these states: active, suspended, & retired. Default search result should include just these versions as well.	TRUE	clInfo	1/1/2010		Grouping	
suspended	Temporarily removed, often done before terminated but not always. New students may not enroll but existing students already enrolled may continue depending on grandfathering rules	TRUE	clInfo	1/1/2010		Released	
retired	Retired/No longer active. Terminated	TRUE	clInfo	1/1/2010		Released	

### Outstanding Issues

- We should consider taking draft and not approved versions out of course version history; they introduce confusion to the user because they are versions based on proposals. One implication of making this change is that there will be missing version numbers in the version list. An alternative might be to give the user the ability to show/hide draft and not approved versions. Be sure to consider potential implications for copy proposal from existing course/proposal.  Seems like a good idea to implement a filter on course version history, but will not be addressed in R1.2. Could look into a sequence number that may only increment on approved versions. **\*\*Leaving this information intact for 2.0 documentation to provide information for future use.**



# CM Feature Matrix 2.0

- General Information
- Definitions
- At-a-Glance: Delivered Learning Experiences and Associated Features
- Feature Matrix
  - Learning Experiences
  - Courses
  - Programs
  - Proposals
  - Course Set
  - Learning Objectives
  - Analysis
    - Course
    - Program
    - Other

## General Information

The KS Curriculum Management Module provides the ability to propose, create, modify and retire learning experiences that are part of an institution's sanctioned curriculum.

## Definitions

Catalog	The collection of approved learning experiences that an institution may offer to its students.
Course	A learning experience that imparts education through a series of activities such as lectures, labs, recitations, etc. within a well-defined time period; it results in an assessment (e.g., grade) and an outcome (e.g., credits).
Credit Course	A course which, upon successful completion, results in the award of academic credit.
Non-Credit Course	A course which, upon successful completion, results in the award of non-academic units, such as Continuing Education Units.
Programs	A prescribed grouping of learning experiences that leads to a recognized body of knowledge; the end result of a completed program may be an acknowledged level of accomplishment (e.g. a major, minor or concentration) or a credential (e.g. certificate, degree, etc.)
Core Program	A prescribed grouping of courses that is required for all students as a foundation of general knowledge. Also known as General Education
Credential Program	A prescribed grouping of learning experiences that leads to a credential.
Baccalaureate	An organized curriculum leading to a baccalaureate degree in a major area of study.
Masters	A postgraduate academic program typically designed for learners after the completion of a baccalaureate program
Doctoral	A postgraduate academic program typically representing the highest level of study in a field; designed for those wishing to conduct research in that field
Professional	A postgraduate academic program typically representing the highest level of study in a field; designed for those wishing to practice in that field
Certificate	An organized curriculum leading to a certificate in an area of study.

Academic Program	aka, Major; An organized curriculum offered as a major area of study recognized as an academic discipline by the higher education community as part of a credential program
Minor	An organized curriculum offered as part of an individual student's credential program and which enhances or complements the credential to be awarded in a manner that leads to specific educational or occupational goals.
Specializations	Variations and/or refinements of an Academic Program that lead to specific educational or occupational goals.
<b>Other Learning Experiences</b>	
Project	An individualized student learning experience designed to culminate in a "product" that synthesizes the learning obtained by the student. Examples of projects include theses, dissertations, research projects, and/or capstone projects. An important attribute of the project is the supervisory committee and associated rules regarding its make-up. Typically, projects will be associated with degree and certificate programs.
Experiential Learning	An individualized student learning experience that emphasizes learning by "doing" rather than "producing." Examples include service learning, internships, clerkships, and co-operative education.
Tests	A procedure for evaluating learning of student; may be used to award credit and/or fulfill a requirement.
<b>Course Set</b>	<b>A set composed of one or more Courses</b>
Fixed Course Set	A Course Set whose elements are defined explicitly by the course number, e.g. (MATH101, MATH102, MATH103)
Dynamic Course Set	A Course Set whose elements are defined implicitly by some criteria, e.g. (100-level math courses); were a new 100-level MATH course be added to the curriculum, it would automatically be added to this course set as well.
Single-Use Course Set	A single-use Course Set is one which is created 'on the fly' when building a rule and is not available for use in other rules.
Reusable Course Set	A reusable Course Set is one which is created prior to building a rule, named, and saved, and is available for use in other rules.
<b>Proposal</b>	<b>A description of a learning experience, typically authored by faculty, that is submitted for review by the appropriate curricular oversight organization; if approved, the learning experience becomes part of the institution's curriculum</b>
<b>Learning Objective</b>	<b>A statement, usually written by a faculty member, that describes the knowledge, skills, or abilities that a student would expect to gain by participating in a learning experience</b>
Single-Use Learning Objectives	Learning Objectives that are defined within the context of a specific Course or Program and are not maintained as part of a centralized inventory.
Governed Learning Objectives	Learning Objectives that are defined by a group of people external (eg. accreditation bodies) or internal (eg. a department, or an organization of biology, chemistry, and physics departments) to the institution. These Learning Objectives may only be modified by the internal organization of people who entered them with the expectation that these centrally managed, 'governed' learning objectives may be imported directly into courses or programs.

## At-a-Glance: Delivered Learning Experiences and Associated Features

	Course	Program*
--	--------	----------

<b>Proposal Processes</b>		
Create via Proposal Process	★	✗
Modify via Proposal Process	★	★
Retire via Proposal Process	★	✗
<b>Administrative Processes</b>		
Create via Administrative Screens	★	★
Modify via Administrative Screens	★	★
Retire via Administrative Screens	★	✗
<b>Versioning</b>		
Modify with Versioning	★	★
Modify without Versioning	★	★
<b>Search/Browse/View</b>		
Search for Proposal	★	★
Search Catalog	★	★
Browse Catalog	★	★
View Catalog	★	★
<b>Learning Objectives</b>		
Attach Learning Objectives	★	★
<b>Sets</b>		
Fixed, Single-Use Sets	★	★

Dynamic and Reusable Sets		
<b>Analyses</b>		
Dependency Analysis		

\*Not all features marked as delivered are available for every program type

Legend	
	Reference implementation delivered via UI in CM 2.0
	No reference implementation delivered in CM 2.0

## Feature Matrix

### Learning Experiences

Type	Supported by Class I Services	Supported by Class II Services
<b>Course</b>		
Credit Course		
Non-Credit Course		
<b>Programs</b>		
Core Program		
Credential Program - Baccalaureate		
Credential Program - Masters		
Credential Program - Doctoral		
Credential Program - Professional		
Credential Program - Certificate		
Academic Program (Major)		
Minor		

Specializations		
Other		
Project		
Experiential Learning		
Tests - Institutionally administered		

#### Legend

- Feature supported/delivered in CM 2.0
- Feature not supported/delivered

#### Definitions

- **Supported by Services:** Feature is supported by Service Design
  - **Supported by Class I Services:** Features is supported by Atomic Services
  - **Supported by Class II Services:** Feature is supported by Composed Services
- **Supported by KS Rice:** Feature supported by KS Rice products KIM and/or KEW
- **Delivered in UI:** A reference implementation of the Feature is delivered in the UI

## Courses

High Level Feature	Feature	Supported by Services	Delivered in UI
Ability to Create a Course	Ability to <b>create</b> a Course through a <b>#proposal process</b>		
	Ability to <b>create</b> a Course through <b>administrative screens</b>		
Ability to Define a Course	Ability to specify key <b>Identifying Information</b>		

	Ability to specify key <b>Descriptive Information</b>	★	★
	Ability to specify alternate identifiers, <b>Cross-Listing</b> and <b>V</b> ersion <b>C</b> odes	★	★
	Ability to specify a <b>Joint Offering</b>	★	★
	Ability to specify key <b>Dates and Terms</b>	★	★
	Ability to identify Course <b>Resour</b> ces - Faculty	★	★
	Ability to identify Course <b>Resour</b> ces - Other	✗	✗
	Ability to define multiple <b>Formats and Activities</b>	★	★
	Ability to define multiple Course <b>Grading and Credit Options</b>	★	★
	Ability to define Course <b>#Learni</b> ng <b>Objectives</b>	★	★
	Ability to define Course <b>Requisit</b> es	★	★
	Ability to specify <b>Organizational Relationships</b>	★	★
	Ability to specify <b>Financial</b> information	★	★
	Ability to attach <b>Supporting Documentation</b>	★	✗
	Ability to attach <b>Comments</b>	★	✗
<b>Ability to Manage a Course</b>	Ability to <b>modify</b> a Course throu <b>gh</b> a <b>#proposal</b> process	★	★
	Ability to <b>modify</b> a Course throu <b>gh</b> administrative screens	★	★
	Ability to <b>retire</b> a Course throu <b>gh</b> a <b>#proposal</b> process	★	★
	Ability to <b>retire</b> a Course throu <b>gh</b> administrative screens	★	★
<b>Ability to Version a Course</b>	Ability to <b>version</b> a Course as a result of a modification	★	★
	Ability <b>not to version</b> a Course as a result of a modification	★	★
<b>Ability to Activate a Course</b>	Ability to <b>activate</b> an approved Course version	★	★
<b>Ability to Publish a Course</b>	Ability to publish approved Course versions in a <b>centralized catalog</b>	★	★
	Ability to designate <b>other publis</b> hing <b>targets</b> (i.e., not the catalog)	★	✗
	Ability to <b>define information</b> to be published by target	★	✗

<b>Ability to Browse for a Course</b>	Ability to <b>browse</b> the catalog for a Course by Subject Area or Organization	★	★
	Ability to <b>browse</b> the catalog for a Course by <b>other attributes</b> (i.e., not Subject Area or Organization)	★	✗
<b>Ability to Search for a Course</b>	Ability to <b>search</b> for a Course	★	★
<b>Ability to View a Course</b>	Ability to <b>view</b> a Course	★	★
	Ability to select from <b>multiple views</b> of a Course (i.e., at-a-glance, detailed and catalog)	★	★
	Ability to <b>view the version history</b> of a Course	★	★
	Ability to <b>print</b> details of a Course	N/A	★
	Ability to <b>export</b> details of a Course	N/A	★

## Programs

Process	Feature	Supported by Services	Delivered in UI
<b>Ability to Create a Program</b>	Ability to <b>create</b> a Program <b>through a #proposal process</b>	★	✗
	Ability to <b>create</b> a Program <b>through administrative screens</b>	★	★
<b>Ability to Define a Program</b>	Ability to specify key Program <b>Id identifying Information</b>	★	★
	Ability to specify key Program <b>Descriptive Information</b>	★	★
	Ability to specify key Program <b>Dates and Terms</b>	★	★
	Ability to identify Program <b>Resources - Faculty</b>	★	★
	Ability to define Program <b>#Learning Objectives</b>	★	★
	Ability to define Program <b>Entrance, Satisfactory Progress and Completion Requirements</b>	★	★
	Ability to specify <b>Organizational Relationships</b>	★	★
<b>Ability to Manage a Program</b>	Ability to <b>modify</b> a Program <b>through a #proposal process</b>	★	★
	Ability to <b>modify</b> a Program <b>through administrative screens</b>	★	★
	Ability to <b>retire</b> a Program <b>through a #proposal process</b>	★	✗
	Ability to <b>retire</b> a Program <b>through administrative screens</b>	★	✗

<b>Ability to Version a Program</b>	Ability to <b>version</b> a Program as a result of a modification	★	★
	Ability <b>not to version</b> a Program as a result of a modification	★	★
<b>Ability to Activate a Program</b>	Ability to <b>activate</b> an approved Program version	★	★
<b>Ability to Publish a Program</b>	Ability to publish approved Program versions in a <b>centralized catalog</b>	★	★
	Ability to designate <b>other publishing targets</b> (i.e., not the catalog)	★	✗
	Ability to <b>define information</b> to be published by target	★	✗
<b>Ability to Browse for a Program</b>	Ability to <b>browse</b> the catalog for a Program	★	★
<b>Ability to Search for a Program</b>	Ability to <b>search</b> for a Program	★	★
<b>Ability to View a Program</b>	Ability to <b>view</b> a Program	★	★
	Ability to <b>view the version history</b> of a Program	★	★
	Ability to <b>print</b> details of a Program	N/A	★
	Ability to <b>export</b> details of a Program	N/A	★

## Proposals

High Level Feature	Feature	Supported by Services	Supported by KS Rice	Delivered in UI
<b>Ability to Start a Proposal</b>	Ability to start a Proposal to <b>create a Course</b> from blank	★	N/A	★
	Ability to start a Proposal to <b>create a Course</b> from a <b>copy</b> of another approved Course	★	N/A	★
	Ability to start a Proposal to <b>create a Course</b> from a <b>copy</b> of another proposed Course	★	N/A	★
	Ability to start a Proposal to <b>create a Course</b> from a <b>template</b>	★	N/A	✗
	Ability to start a Proposal to <b>modify a Course</b> from a copy of the Course	★	N/A	★
	Ability to start a Proposal to <b>modify a Program</b> from a copy of the Program	★	N/A	★
<b>Ability to Save a Proposal</b>	Ability to <b>save and return</b> to a Proposal	★	N/A	★

	Ability to <b>cancel</b> a Proposal	★	N/A	★
<b>Ability to Collaborate on a Proposal</b>	Ability to attach <b>Supporting Documents</b> to a Proposal	★	N/A	★
	Ability to add Collaborators to <b>View, Comment and/or Edit</b> a Proposal	N/A	★	★
	Ability to <b>notify</b> Collaborators that their participation is requested	N/A	★	★
	Ability to <b>remove</b> Collaborators from a Proposal	N/A	★	★
<b>Ability to Review a Proposal</b>	Ability to specify a workflow based on <b>Proposal type</b>	N/A	★	★
	Ability to specify a workflow based on <b>Proposal data</b>	N/A	★	✗
	Ability to <b>submit</b> a Proposal to workflow	N/A	★	★
	Ability to <b>withdraw</b> a Proposal from workflow	N/A	★	★
	Ability to assign reviewers <b>View, Comment and/or Edit</b> proposal permissions	N/A	★	★
	Ability to <b>Request a Decision</b> from a reviewer	N/A	★	★
	Ability to <b>notify</b> reviewers of a pending review/decision request	N/A	★	★
	Ability of reviewers to <b>Enter a Decision</b> on the proposal	N/A	★	★
	Ability of reviewers to determine the <b>status</b> of a proposal	N/A	★	★
	Ability to view the <b>History</b> of the proposal as it moves through the review process	★	✗	✗
	Ability to record the final <b>Outcome</b> of the review process	★	★	★
<b>Ability to Group Proposals</b>	Ability to <b>group</b> proposals	✗	N/A	✗
	Ability to <b>review grouped proposals</b> simultaneously	✗	N/A	✗
<b>Ability to Archive a Proposal</b>	Ability to <b>archive</b> a proposal	★	N/A	✗

<b>Search for a Proposal</b>	Ability to <b>Search</b> for Proposals	★	N/A	★
<b>Ability to View a Proposal</b>	Ability to <b>view</b> a Proposal	★	N/A	★
	Ability to <b>print</b> a Proposal	N/A	N/A	★
	Ability to <b>export</b> details of a Program	N/A	N/A	★

## Course Set

High Level Feature	Feature	Supported by Services	Delivered in UI
<b>Create a Course Set</b>	Ability to create a <b>fixed or dynamic</b> Course Set	★	★
	Ability to create a <b>single-use and/or reusable</b> Course Set	★	★
	Ability to create Course Set that contains <b>proposed or approved</b> courses	★	★
	Ability to create a Course Set that contains <b>courses and/or other Course Sets</b>	★	★
<b>Manage a Reusable Course Set</b>	Ability to <b>modify</b> reusable Course Sets	★	★
	Ability to <b>retire</b> reusable Course Sets	★	✗
	Ability to <b>delete</b> reusable Course Sets	★	✗
	Ability to <b>version</b> reusable Course Sets	✗	✗

## Learning Objectives

High Level Feature	Feature	Supported by Services	Delivered in UI
<b>Ability to Define Learning Objectives</b>	Ability to <b>author</b> Learning Objectives	★	★
	Ability to <b>edit</b> Learning Objectives	★	★
	Ability to <b>delete</b> Learning Objectives	★	★
	Ability to <b>search for, copy and then edit</b> Learning Objectives	★	★
<b>Ability to Organize Learning Objectives</b>	Ability to <b>order</b> Learning Objectives	★	★
	Ability to create a <b>hierarchy</b> of Learning Objectives with levels and sublevels	★	★
	Ability to <b>create</b> Learning Objectives <b>categories</b>	★	★

	Ability to maintain an <b>inventory</b> of Learning Objectives <b>categories</b>	★	★
	Ability to manage an inventory of <b>Governed Learning Objectives</b>	✗	✗
	Ability to <b>categorize</b> Learning Objectives	★	★
<b>Ability to Attach Learning Objectives</b>	Ability to attach Learning Objectives to a <b>Course</b>	★	★
	Ability to attach Learning Objectives to a <b>Course Activity</b>	★	✗
	Ability to attach Learning Objectives to a <b>Program</b>	★	★
<b>Ability to Map Learning Objectives</b>	Ability to <b>map</b> Program Learning Objectives to Courses that support them	✗	✗

## Analysis

### Course

High Level Feature	Feature	Supported by Services	Delivered in UI
<b>Ability to determine the Dependence of Other Courses on a Selected Course</b>	Ability to determine which Courses are <b>Jointly-Offered</b> with the Selected Course	★	★
	Ability to determine which Courses are <b>Cross-listed</b> with the Selected Course	★	★
	Ability to determine which Courses use the Selected Course in a <b>Prerequisite Rule</b>	★	★
	Ability to determine which Courses list the Selected Course in a <b>Corequisite Rule</b>	★	★
	Ability to determine which Courses list the Selected Course in a <b>Antirequisite Rule</b>	★	★
	Ability to determine which Courses list the Selected Course as a <b>Recommended Preparation</b>	★	★
	Ability to determine <b>Oversight Organization(s)</b> of the Dependent Courses	★	★
<b>Ability to determine the Dependence of Programs on a Selected Course</b>	Ability to determine which Programs list the Selected Course as an <b>Entrance Requirement</b>	★	★
	Ability to determine which Programs list the Selected Course as a <b>Satisfactory Progress Requirement</b>	★	★

	Ability to determine which Programs list the Selected Course as a <b>Completion Requirement</b>	★	★
	Ability to determine <b>Oversight Organization(s)</b> of the Dependent Programs	★	★
<b>Ability to determine the membership of a Selected Course in Reusable Course Sets</b>	Ability to determine which Reusable <b>Fixed Course Sets</b> contain the Selected Course	★	★
	Ability to determine which Reusable <b>Dynamic Course Sets</b> contain the Selected Course	★	✗
	Ability to determine <b>Oversight Organization(s)</b> of the Dependent Course Sets	★	✗
<b>Ability to access details of the Dependencies</b>	Ability to <b>filter</b> the analysis results	N/A	★
	Ability to <b>view</b> the Dependent Course from the analysis results	★	★
	Ability to <b>view</b> the Dependent Program from the analysis results	★	★
	Ability to <b>view</b> the Dependent Course Set from the analysis results	★	★

## Program

High Level Feature	Feature	Supported by Services	Delivered in UI
<b>Ability to determine the Dependencies within a Selected Program</b>	Ability to view which Courses associated with the Selected Program are <b>not owned by the same Oversight Organization</b> as the Selected Program	★	✗
	Ability to determine the <b>Prerequisite chains for required Courses</b> in a Selected Program	✗	✗
	Ability to run an analysis of a Select Program based on <b>Learning Objectives</b>	✗	✗
<b>Ability to determine the Dependencies across Selected Programs</b>	Ability to view which Courses are <b>shared across two or more Selected Programs</b>	★	✗

## Other

High Level Feature	Feature	Supported by Services	Delivered in UI
<b>Ability to print/export the results of a Dependency Analysis</b>	Ability to <b>print</b> the results of a Dependency Analysis	N/A	★

	Ability to <b>export</b> the results of a Dependency Analysis	N/A	
<b>Ability to require a Dependency Analysis as part of a Modify or Retire Course Proposal</b>	Ability to <b>automatically trigger</b> a Dependency Analysis as part of a Modify or Retire Course Proposal		
	Ability to <b>require</b> a Dependency Analysis be run and attached to a a Modify or Retire Course Proposal <b>at a preconfigured node in the workflow</b>		
	Ability to send <b>notifications</b> to Oversight Organization(s) of Dependent Courses, Course Sets, and Programs		



# KS Curriculum Management 2.0 QuickStart Installation Guide

## Need Help?

If you have any questions about troubleshooting your installation, please join our [Implementation Users Group](#) list serv. Information about the IUG can be found [here](#). Once joined, you can access our [forum archives](#) as well as ask questions as you have them.

- [Introduction](#)
- [1. Prerequisite Knowledge](#)
- [2. Set Up the Database](#)
  - [Install the Oracle JDBC Driver in maven's local repository](#)
  - [Load the Kuali Student Bundled Data](#)
- [3. Download and Configure Kuali Student Curriculum Management 2.0](#)
  - [Download the WAR File:](#)
  - [Create Property File\(s\) for WAR Configuration](#)
    - [Supported Properties for ks-with-rice-bundled](#)
    - [Configuring CAS with KS Curriculum Management \(Bundled Rice\)](#)
- [4. Deploying the Application in Tomcat](#)
- [5. Access the Application](#)
- [6. Source Code](#)

## Introduction

The goal of this guide is to get you up and running with the reference implementation of KS CM 2.0 with minimal steps and overhead following the **KS-Bundled deployment strategy**. In this deployment strategy, all Kuali Student modules and a preconfigured implementation of Kuali Rice are in a single WAR file. This deployment lends itself well to developers and analysts who want to get something up quickly for customization, testing and analysis. This setup will get a server running but is not intended as the starting point for implementation but as a way to set up a local sandbox for evaluation purposes.

If this deployment is intended for enterprise use, the community recommends following the **KS-Distributed deployment strategy** where all Kuali Student modules are in one WAR file and a preconfigured Kuali Rice implementation resides in a separate WAR file. For information on the distributed deployment, please see the [KS CM 2.0 Implementation and Deployment Guide](#).

For more on deployment strategies, see [Implementation and Deployment Overview](#).

## 1. Prerequisite Knowledge

Familiarity with the following technologies is a necessary prerequisite for installing and deploying Kuali Student Curriculum Management.

- Source code management using [Subversion](#)
- Building and assembling of projects using [Maven](#)
- Deploying WAR files using [Tomcat](#)

## 2. Set Up the Database

The Curriculum Management database extracts are provided as one combined schema. To setup the database, the following tools are required:

- **Subversion 1.7.x:** See <http://subversion.apache.org/packages.html> to download a Subversion client
- **Oracle or Oracle XE:** You will need to install version 10g or later of Oracle or Oracle XE (Oracle Express) locally or be given the password for the Oracle user "SYS". The Oracle user "SYS" has the permissions necessary for setting up the Kuali Student database.
- **JDK (Java Developer Kit) 6**



JDK 7 is not supported at this time

- **Maven 3.0.4 or later**

## Install the Oracle JDBC Driver in maven's local repository

Oracle licenses their JDBC driver in a way that prevents it from being publicly available. You will need to install the jar in your local Maven repository so that it is available to Maven when loading data.

1. Obtain a copy of ojdbc14.jar
  - The correct version to use is available [here](#) or you can download it from [Oracle](#) (requires creating an account on OTN)

- If you have installed [Oracle Express Edition](#) locally, this jar file is already on your local system at:

```
/oraclexe/app/oracle/product/10.2.0/server/jdbc/lib/ojdbc14.jar
```

2. Use the [Maven Install Plugin](#) to copy ojdbc14.jar into your local maven repository.

```
mvn install:install-file -DgroupId=com.oracle -DartifactId=ojdbc14
-Dversion=10.2.0.3.0 -Dpackaging=jar -Dfile=ojdbc14.jar
```

If the jar is installed correctly, you should see output similar to the following:

```
[INFO] Installing ojdbc14.jar to
./.m2/repository/com/oracle/ojdbc14/10.2.0.3.0/ojdbc14-10.2.0.3.0.jar
```

## Load the Kuali Student Bundled Data

The ks-bundled-db module contains the learning unit management database and the Rice database with KS Curriculum Management configuration and workflow data.

1. Checkout the data sets module for the 2.0.2 tag:

```
svn co
http://svn.kuali.org/repos/student/enrollment/ks-deployments/tags/ks-deployments-
2.0.2-cm/ks-cfg-dbs/
```

2. Change to the directory for the bundled data set:

```
cd ks-cfg-dbs/ks-bundled-db
```

3. Review the Maven Impex properties listed in the table below. At a minimum, you will need to supply a value for the `ks.impex.dba.password` property in the next step.

Property	Default Value	Description
<code>ks.impex.url</code>	<code>jdbc:oracle:thin:@localhost:1521:XE</code>	The JDBC url for the database. Override as needed.
<code>ks.impex.dba.username</code>	SYS AS SYSDBA	Do not override! This is a specific Oracle DBA user with the permissions needed to create/drop Oracle schemas and grant the permissions needed for Kuali Student.
<code>ks.impex.dba.password</code>	<code>change_on_install</code>	Default password for the SYS user when Oracle is installed. Override this with the password for the SYS user for your Oracle instance
<code>ks.impex.username</code>	KSBU NDLED	The Oracle user the Kuali Student application will be connecting to the database as.
<code>ks.impex.password</code>	KSBU NDLED	The password for the Oracle user the Kuali Student application will be connecting as.
<code>ks.impex.driver</code>	<code>oracle.jdbc.driver.OracleDriver</code>	JDBC driver for Oracle.

ks.impex.sql.drop	<pre> BEGIN EXECUTE IMMEDIATE 'DROP USER \${ks.impex.username} CASCADE'; EXCEPTION WHEN OTHERS THEN     IF SQLCODE != -01918 THEN         RAISE;     END IF; END; </pre>	SQL to drop an Oracle user. This drops the user as well as any objects owned by the user (tables, indexes, sequences, views, etc). This ensures that the Impex process is repeatable by guaranteeing that any previously created data has been removed and that the Impex process will be starting with an empty database. Ignore the error if the user does not exist. Execution of this SQL is tied to the Maven ""clean"" phase by default.
ks.impex.sql.validate	SELECT SYSDATE FROM DUAL	Simple query for validating a JDBC connection. Execution of this SQL is tied to the Maven "validate" phase by default.

ks.impex.sql.initialize

```
CREATE USER
${ks.impex.username}
}
IDENTIFIED BY
${ks.impex.password}
}
DEFAULT TABLESPACE
users
TEMPORARY
TABLESPACE temp
QUOTA unlimited ON
users
/
GRANT CREATE
PROCEDURE
, CREATE SEQUENCE
, CREATE SESSION
, CREATE SYNONYM
, CREATE TABLE
, CREATE TRIGGER
, CREATE TYPE
, CREATE VIEW
TO
${ks.impex.username}
}
/
GRANT SELECT ON
pending_trans$ to
${ks.impex.username}
}
/
GRANT SELECT ON
dba_2pc_pending to
${ks.impex.username}
}
/
GRANT SELECT ON
dba_pending_transactions to
${ks.impex.username}
}
/
GRANT EXECUTE ON
dbms_system to
${ks.impex.username}
}
/
```

SQL for creating an Oracle user and granting the permissions needed for Kuali Student. Execution of this SQL is tied to the Maven ""initialize"" phase by default.

4. Validate that Maven can connect to your Oracle instance as follows:

```
mvn validate -Pks-db,oracle -Dks.impex.url=jdbc:oracle:thin:@<db
host>:<port>:<SID> -Dks.impex.dba.password=<SYS password>
```

If validation fails with the error "com.oracle.jdbc14.jar:10.2.0.3.0", make sure you have installed the JDBC driver as described in section 1.2 Install the Oracle JDBC Driver above. A successful result is indicated by output similar to the following:

```

[INFO] --- maven-antrun-plugin:1.1:run (dba-config) @ ks-bundled-db ---
[INFO] Executing tasks
[echo] JDBC Url - jdbc:oracle:thin:@localhost:1521:XE
[echo] JDBC Driver - oracle.jdbc.driver.OracleDriver
[echo] Username - xxxxxxxx
[echo] Password - xxxxxxxx
[echo] SQL: 'SELECT SYSDATE FROM DUAL'
[INFO] Executed tasks
[INFO]
[INFO] --- sql-maven-plugin:1.4:execute (validate-dba-config) @ ks-bundled-db ---
[INFO] Executing commands
[INFO] 1 of 1 SQL statements executed successfully
[INFO] -----
[INFO] BUILD SUCCESS

```

- Load the dataset as follows:

```

mvn clean install -Pks-db,oracle -Dks.impex.url=jdbc:oracle:thin:@<db
host>:<port>:<SID> -Dks.impex.dba.password=<SYS password>

```

### 3. Download and Configure Kuali Student Curriculum Management 2.0

These instructions explain how to download a single WAR file to install all Kuali Student modules and a preconfigured implementation of Kuali Rice. If instead you would like to deploy the Kuali Student modules and Kuali Rice from separate WAR files in distributed fashion, please see the [KS CM 2.0 Implementation and Deployment Guide](#).

#### Download the WAR File:

For the bundled configuration, download the file:

- <http://shrub.appspot.com/maven.kuali.org/release/org/kuali/student/web/ks-with-rice-bundled/2.0.2-cm/ks-with-rice-bundled-2.0.2-cm.war>

#### Create Property File(s) for WAR Configuration

You will need to create a property file for the WAR file. You will not need to modify the WAR to configure the application. By default the application will look for the property files in the following location:

- `${user.home}/kuali/main/${environment}/ks-with-rice-bundled-config.xml`

where

- `${user.home}` will be the home directory of the user the server runs as.
- `${environment}` defaults to `dev`, but can be changed with a system parameter.

If it is not possible to store the property file in the user's home directory, an alternate location can be defined by setting a system parameter named `additional.config.locations`. The filename should be included--for example, `-Dadditional.config.locations=/custom/location/config_file_name.xml`

Use the following format for the property file (`ks-with-rice-bundled-config.xml` config file is shown in this example) making sure to substitute the proper values for Oracle SID, `keystore.file` and the datasource username and password (if you chose non-default values for `ks.impex.username` in the previous impex loading step):

### ks-with-rice-bundled-config.xml example

```
<config>
    <param name="appserver.url">https://my.url.org</param>
    <param name="app.context.name">ks-with-rice-bundled</param>

    <param name="datasource.url">jdbc:oracle:thin:@my.url.org:1521:SID</param>
    <param name="datasource.username">KSBUNDLED</param>
    <param name="datasource.password">KSBUNDLED</param>

    <param name="keystore.file">/some/file/system/path/rice.keystore</param>
</config>
```

You can obtain a copy of the `rice.keystore` file to place in a path in your file system from [here](#) or from within the war at `WEB-INF/classes/rice.keystore`.

### Supported Properties for ks-with-rice-bundled

**NOTE:** When using the `ks-with-ricebundled` module, the settings for `ks.lum.datasource.*` and `ks.core.datasource.*` default to the values of `datasource.*`

Property	Description	Use
<code>appserver.url</code>	This is the url for the root of the appserver (does not include the context name). ex: <code>http://my.school.edu:1234</code>	required
<code>app.context.name</code>	The name of deployed context	required
<code>keystore.file</code>	The file full path to the rice keystore file, currently located in the WAR <code>WEB-INF/classes/rice.keystore</code>	required
<code>datasource.url</code>	The jdbc connection string for the rice datasource	required
<code>datasource.username</code>	username for rice datasource	required
<code>datasource.password</code>	password for rice datasource	required
<code>ks.lum.datasource.url</code>	jdbc connection string for the lum datasource	optional
<code>ks.lum.datasource.username</code>	lum datasource username	optional
<code>ks.lum.datasource.password</code>	lum datasource password	optional
<code>ks.core.datasource.url</code>	jdbc connection string for the core datasource	optional
<code>ks.core.datasource.username</code>	core datasource username	optional
<code>ks.core.datasource.password</code>	core datasource password	optional
<code>log4j.settings.path</code>	This changes the log4j config file that is used, should include the filename. The default log4j settings are in the project <a href="#">here</a>	optional
<code>ks.core.datasource.minSize</code>	Set the minimum db pool size for the core datasource.	optional
<code>ks.core.datasource.maxSize</code>	Set the maximum db pool size for the core datasource.	optional
<code>ks.core.jpa.showSql</code>	Set hibernate statement logging for the core datasource (true or false).	optional

ks.lum.datasource.minSize	Set the minimum db pool size for the lum datasource.	optional
ks.lum.datasource.maxSize	Set the maximum db pool size for the lum datasource.	optional
ks.lum.jpa.showSql	Set hibernate statement logging for the lum datasource (true or false).	optional
datasource.minSize	Set the minimum db pool size for the rice datasource.	optional
datasource.maxSize	Set the maximum db pool size for the rice datasource.	optional
additional.config.locations	define location of additional configuration parameter files (use a system parameter)	optional
ks.institutional.context	defines the location for a spring context file which can be used to override or add beans/customizations	optional
ks.authentication.context	sets the spring context to be used to configure spring security	optional
ks.default.security.cas.serverAddress	set the url of your CAS server if using one	optional

## Configuring CAS with KS Curriculum Management (Bundled Rice)

To configure KS Curriculum Management to use your CAS server for authentication, set the following properties:

```
<param name="ks.default.security.cas.serverAddress">CASLoginURL</param>
<param name="ks.authentication.context">classpath:ks-spring-security-cas.xml</param>
```

where **CASLoginURL** is the URL for your CAS server login (for example, \*<https://login.ks.edu/cas>\*

If your CAS server is using the Proxy Mechanism or if you want to enable it, set the following property:

```
<param name="ks.default.security.cas.useCasProxyMechanism">true</param>
```

**Note:** You must configure the server for SSL if you want to use the Proxy Mechanism.

## 4. Deploying the Application in Tomcat

To deploy the application in Tomcat perform the following steps (if you are not using Tomcat as the application server, you will need to perform appropriate steps pertinent to that server).

1. Add the following memory settings to your server configuration

```
-Xmx1024m -XX:PermSize=256m
```

*For Unix, add these settings to CATALINA\_OPTS environment variable for the user starting tomcat.*

2. Add the Oracle JAR to your Tomcat server's classpath (typically done by copying the JAR in `TOMCAT_HOME/lib` directory).
3. Copy the WAR file into `TOMCAT_HOME/webapps` (it is recommended that the version number be removed from the war filename before copying).
4. Start Tomcat

## 5. Access the Application

To access the application, go to the `appserv.url` that you specified in your property file.

You should be presented with the login screen. By default, the administrative user name is "admin" and the password is "admin".

**NOTE:** An index page will be available at <http://yoururl.edu/warname/> where `warname` is the name of the WAR file you deployed.

To find other predefined users, go into Rice, click the Administration tab, then click the person link to search for usernames. The default password for all users is the username.

## **6. Source Code**

The source code for KS Curriculum Management 2.0 releases can be download [here](#).



## **KS CM 2.0 Implementation and Deployment Guide**

(CM 2.0) 1. Implementation and Deployment Overview .....	3
(CM 2.0) Distributed Versus Bundled Kuali Rice .....	3
(CM 2.0) 2. Setting up the Development Environment .....	3
(CM 2.0) 3. Installing KS Curriculum Management .....	4
(CM 2.0) 3.1 Setting up the Database .....	4
(CM 2.0) 3.1.1 Upgrading the Database .....	4
(CM 2.0) 3.1.2 Install the Oracle JDBC Driver .....	5
(CM 2.0) 3.1.3 Load the Kuali Student Application Data .....	5
(CM 2.0) 3.1.4 Load the Kuali Student Rice Data .....	7
(CM 2.0) 3.2 Downloading and Configuring KS Curriculum Management .....	7
(CM 2.0) 3.2.1 Download the WAR Files .....	8
(CM 2.0) 3.2.2 Create Property Files for WAR Configuration .....	8
(CM 2.0) 3.2.3 Supported Properties for ks-rice-standalone .....	11
(CM 2.0) 3.2.4 Supported Properties for ks-with-rice-embedded .....	12
(CM 2.0) 3.3 Deploying the Application in Tomcat .....	13
(CM 2.0) 3.4 Accessing the Application .....	14
(CM 2.0) 4. Configuring KS Curriculum Management .....	14
(CM 2.0) 4.1 Other Deployment Considerations .....	14
(CM 2.0) 4.1.1 The Service Bus .....	14

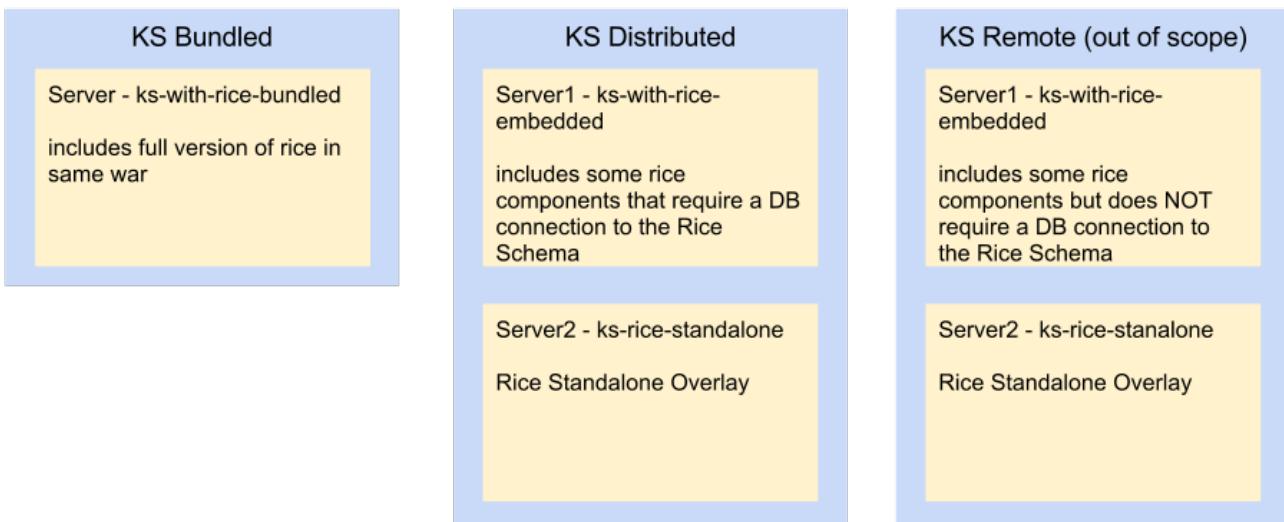
# KS CM 2.0 Implementation and Deployment Guide

## (CM 2.0) 1. Implementation and Deployment Overview

This document provides details for deploying Kuali Student Curriculum Management 2.0, including database preparation and system installation. Summary information is also provided for setting up the development environment before installation and configuring the system after installation, but details are provided in the following documents:

- [Kuali Student Developer Guide](#) – The Developer Guide provides detail on setting up the standard development environment for deploying Kuali Student at your institution, including the Java Development Kit (JDK), Eclipse, and Maven.
- [Kuali Student Curriculum Management 2.0 Configuration Guide](#) – The Configuration Guide provides details on how to configure KS Curriculum Management including the user interface, data, business rules and processes, enumerations and searches.

## KS Deployment Strategies



## (CM 2.0) Distributed Versus Bundled Kuali Rice

Kuali Student Curriculum Management supports two deployment configurations: bundled and distributed. **This document covers the Distributed configuration only.**

- **Distributed Configuration**
  - This configuration consists of two application servers. All Kuali Student modules are in one WAR file deployed to one application server and a preconfigured Kuali Rice implementation in a separate WAR file deployed to another application server.
- **Bundled Configuration**
  - This configuration consists of a single application server. All Kuali Student modules and a preconfigured implementation of Kuali Rice contained in a single WAR file. Installation is much simpler in this configuration, but it may not meet your deployment preferences. Instructions for installing the bundled configuration are detailed in the [QuickStart Installation Guide](#).

## (CM 2.0) 2. Setting up the Development Environment

Kuali Student uses the standard tools identified by the Kuali Foundation to promote ease of code sharing and consistency in the Kuali code base. Table 1 lists the tools and frameworks used for development of the Kuali Student Curriculum Management.

Tool/Framework	Description	License
Sun JDK (Java Development Kit)	Programming tools for java developers, including a loader, compiler, debugger, etc.	GNU General Public License
Apache Tomcat	A JSP (JavaServer Pages) servlet container.	Apache License
Eclipse and Eclipse Plug-ins	The software development environment and plug-ins for integration with SVN and Maven.	Eclipse Public License
Apache SVN (Subversion)	A software versioning and revision control system for managing source code.	Apache License
Spring Framework	A Java application framework for data access and Model-View-Controller (MVC) framework. Spring provides the means to configure Kuali Student for institutional needs.	Apache License

**Table 1: Standard Development Environment for Kuali Student Curriculum Management**

You will need to set up this standard environment as described in the [developer guide](#) if you intend to configure Kuali Student and/or contribute code back to the project.

## (CM 2.0) 3. Installing KS Curriculum Management

Before you begin, please review the [KS Curriculum Management 2.0 Release Notes](#). You may also find the other documentation listed on the [KS Curriculum Management 2.0 Documentation](#) page useful.

### (CM 2.0) 3.1 Setting up the Database

The Curriculum Management database extracts are provided as one combined schema. To setup the database, the following tools are required:

- **Subversion 1.7.x:** Version 1.7.x is recommended. See <http://subversion.apache.org/packages.html> to download.
- **Oracle or Oracle XE:** You will need to install version 10g or later of Oracle or Oracle XE (Oracle Express) locally or be given access to an Oracle user account that has the necessary permissions for creating tables, views, indexes, and sequences.
- **JDK (Java Developer Kit) 6 (JDK 7 is not supported at this time)**
- **Maven 3.0.4 or later**

#### (CM 2.0) 3.1.1 Upgrading the Database

##### Setup

A set of scripts has been created to upgrade an existing database schema from CM 1.2.1 to work with CM 2.0. These scripts will update table structures and add or alter bootstrap data that is required by CM 2.0.

[Download the scripts](#)

##### SQL Execution

The instructions below are a reference to use SQL\*Plus as a tool to execute the upgrade scripts, though there are many other tools that could be used for the same purpose.

The syntax to use is:

```
sqlplus <username>@XE @<filename>
```

Where <username> is the database user, and <filename> is the name of the file to execute.

SQLPlus should start and ask for a password. If you don't want to enter the password for every execution, you can use the command line syntax:

```
sqlplus <username>/<password>@XE @<filename>
```

Simply run this command for each sql file in the attached set of files. They need to be run in the order they are listed (by the date prefix for each file).

### (CM 2.0) 3.1.2 Install the Oracle JDBC Driver

Oracle licenses their JDBC driver in a way that prevents it from being publicly available. You will need to install the jar in your local Maven repository so that it is available to Maven when loading data.

1. Obtain a copy of ojdbc14.jar
  - The correct version to use is available [here](#) or you can download it from [Oracle](#) (requires creating an account on OTN)
  - If you have installed [Oracle Express Edition](#) locally, this jar file is already on your local system at:

```
• /oraclexe/app/oracle/product/10.2.0/server/jdbc/lib/ojdbc14.jar
```

2. Use the Maven Install Plugin (see <http://maven.apache.org/plugins/maven-install-plugin/>) to copy ojdbc14.jar into your local maven repository (the following command is all one line).

```
3. mvn install:install-file -DgroupId=com.oracle -DartifactId=ojdbc14  
-Dversion=10.2.0.3.0 -Dpackaging=jar -Dfile=ojdbc14.jar
```

If the jar is installed correctly, you should see output similar to the following:

```
[INFO] Installing ojdbc14.jar to  
/m2/repository/com/oracle/ojdbc14/10.2.0.3.0/ojdbc14-10.2.0.3.0.jar
```

### (CM 2.0) 3.1.3 Load the Kuali Student Application Data

The ks-app-db module contains the learning unit management database and the Rice database with KS Curriculum Management configuration and workflow data.

1. Checkout the data sets module for the 2.0.2 tag:

```
svn co  
https://svn.kuali.org/repos/student/enrollment/ks-deployments/tags/ks-deployments  
-2.0.2-cm/ks-cfg-dbs/
```

1. Change to the directory for the application data set:

```
cd ks-cfg-dbs/ks-app-db
```

2. Review the Maven Impex properties listed in *Table 1: Maven Impex Properties for KS Application DB* below. At a minimum, you will need to supply a value for the **ks.impex.dba.password** property in the next couple of steps.
3. Validate that Maven can connect to your Oracle instance as follows (command is all one line):

```
mvn validate -Pks-db,oracle -Dks.impex.url=jdbc:oracle:thin:@<db  
host>:<port>:<SID> -Dks.impex.dba.password=<SYS password>
```

If validation fails with the error “com.oracle:ojdbc14:jar:10.2.0.3.0”, make sure you have installed the JDBC driver as described in section 3.1.2: [Install the Oracle JDBC Driver](#).

4. Load the dataset as follows (command is all one line):

```
mvn clean install -Pks-db,oracle -Dks.impex.url=jdbc:oracle:thin:@<db
host>:<port>:<SID> -Dks.impex.dba.password=<SYS password>
```

**Table 1: Maven Impex Properties for KS Application DB**

Property	Default Value	Description
ks.impex.url	jdbc:oracle:thin:@localhost:1521:XE	The JDBC url for the database. Override as needed.
ks.impex.dba.username	SYS AS SYSDBA	<b>Do not override!</b> This is a specific Oracle DBA user with the permissions needed to create/drop Oracle schemas and grant the permissions needed for Kuali Student.
ks.impex.dba.password	change_on_install	Default password for the SYS user when Oracle is installed. Override this with the password for the SYS user for your Oracle instance
ks.impex.username	KSAPP or KSRICE	The Oracle user the Kuali Student application will be connecting to the database as. Use the value appropriate for the dataset you are loading.
ks.impex.password	KSAPP or KSRICE	The password for the Oracle user the Kuali Student application will be connecting as. Use the value appropriate for the dataset you are loading.
ks.impex.driver	oracle.jdbc.driver.OracleDriver	JDBC driver for Oracle.
ks.impex.sql.drop	BEGIN EXECUTE IMMEDIATE 'DROP USER \${ks.impex.username} CASCADE'; EXCEPTION WHEN OTHERS THEN IF SQLCODE != -01918 THEN RAISE; END IF;END;	SQL to drop an Oracle user. This drops the user as well as any objects owned by the user (tables, indexes, sequences, views, etc). This ensures that the Impex process is repeatable by guaranteeing that any previously created data has been removed and that the Impex process will be starting with an empty database. Ignore the error if the user does not exist. Execution of this SQL is tied to the Maven "clean" phase by default.
ks.impex.sql.validate	SELECT SYSDATE FROM DUAL	Simple query for validating a JDBC connection. Execution of this SQL is tied to the Maven "validate" phase by default.

ks.impex.sql.initialize	<pre> CREATE USER \${ks.impex.username} IDENTIFIED BY \${ks.impex.password} DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp QUOTA unlimited ON users / GRANT CREATE PROCEDURE , CREATE SEQUENCE , CREATE SESSION , CREATE SYNONYM , CREATE TABLE , CREATE TRIGGER , CREATE TYPE , CREATE VIEW TO \${ks.impex.username} / GRANT SELECT ON pending_trans\$ to \${ks.impex.username} / GRANT SELECT ON dba_2pc_pending to \${ks.impex.username} / GRANT SELECT ON dba_pending_transactions to \${ks.impex.username} / GRANT EXECUTE ON dbms_system to \${ks.impex.username} / </pre>	SQL for creating an Oracle user and granting the permissions needed for Kuali Student. Execution of this SQL is tied to the Maven "initialize" phase by default.
-------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------

### (CM 2.0) 3.1.4 Load the Kuali Student Rice Data

The ks-rice-db module contains the Rice database only (Organization, Document, etc.).

- Checkout the data sets module for the 2.0.2 development stream:

```

svn co
https://svn.kuali.org/repos/student/enrollment/ks-deployments/tags/ks-deployments
-2.0.2-cm/ks-cfg-dbs/

```

- Change to the directory for the Rice data set:

```

cd ks-cfg-dbs/ks-rice-db

```

- Review the Maven Impex properties listed in *Table 1: Maven Impex Properties for KS Application DB* found [here](#). At a minimum, you will need to supply a value for the **ks.impex.dba.password** property in the next couple of steps.
- Validate that Maven can connect to your Oracle instance as follows (command is all one line):

```

mvn validate -Pks-db,oracle -Dks.impex.url=jdbc:oracle:thin:@<db
host>:<port>:<SID> -Dks.impex.dba.password=<SYS password>

```

If validation fails with the error “com.oracle.ojdbc14.jar:10.2.0.3.0”, make sure you have installed the JDBC driver as described in section [3.1.2: Install the Oracle JDBC Driver](#).

- Load the dataset as follows (command is all one line):

```

mvn clean install -Pks-db,oracle
-Dks.impex.url=jdbc:oracle:thin:@<dbhost>:<port>:<SID>
-Dks.impex.dba.password=<SYS password>

```

## (CM 2.0) 3.2 Downloading and Configuring KS Curriculum Management

Kuali Student Curriculum Management supports two deployment configurations: bundled and distributed. This document covers the Distributed configuration only.

- **Distributed Configuration (2 servlet containers)**
  - This configuration consists of two WAR files
    - All Kuali Student modules are in one WAR file and
    - a preconfigured Kuali Rice implementation in a separate WAR file.
- **Bundled Configuration (single servlet container)**
  - All Kuali Student modules and a preconfigured implementation of Kuali Rice contained in a single WAR file. Installation is much simpler in this configuration, but it may not meet your deployment preferences. Instructions for installing the bundled configuration are detailed in the [QuickStart Installation Guide](#).

### (CM 2.0) 3.2.1 Download the WAR Files

For the **distributed** configuration, you will need the following files:

<http://shrub.appspot.com/maven.kuali.org/release/org/kuali/student/web/ks-rice-standalone/2.0.2-cm/ks-rice-standalone-2.0.2-cm.war>  
contains a preconfigured Kuali Rice implementation.

[shrub.appspot.com/maven.kuali.org/release/org/kuali/student/web/ks-with-rice-embedded/2.0.2-cm/ks-with-rice-embedded-2.0.2-cm.war](http://shrub.appspot.com/maven.kuali.org/release/org/kuali/student/web/ks-with-rice-embedded/2.0.2-cm/ks-with-rice-embedded-2.0.2-cm.war)  
contains all Kuali Student modules (with embedded Kuali Rice components).

### (CM 2.0) 3.2.2 Create Property Files for WAR Configuration

You will need to create a property file for each WAR file. You will not need to modify the WAR to configure the application. By default the applications will look in the following places for the property files.

- \${user.home}/kuali/main/\${environment}/ks-rice-standalone-config.xml
- \${user.home}/kuali/main/\${environment}/ks-with-rice-embedded-config.xml

where

- \${user.home} is the home directory of the username the server will run as
- \${environment} defaults to **dev**, but can be changed with a system parameter.

If it is not possible to store the property file in the user's home directory, an alternate location can be defined by setting a system parameter named `additional.config.locations` and including the filename (for example, `-Dadditional.config.locations=/custom/location/config_file_name.xml`)

Find quoted below two example files with ks-rice-stand-alone WAR running on appserv-1.ks.kuali.net on port 80 and ks-with-rice-embedded WAR running on appserv-2.ks.kuali.net on port 80.

### **ks-rice-standalone-config.xml example**

```
<config>
    <!-- Please fill in proper values for these 2 parameters! -->
    <param name="appserver.url">http://appserv-1.ks.kuali.net</param>
    <param name="app.context.name">ks-rice-standalone</param>

    <!-- Oracle example -->
    <param name="datasource.obj.platform">Oracle9i</param>
    <param name="datasource.url">jdbc:oracle:thin:@dbaddress:1521:SID</param>
    <param name="datasource.username">KSRICE</param>
    <param name="datasource.password">KSRICE</param>
    <param name="datasource.driver.name">oracle.jdbc.driver.OracleDriver</param>
    <param name="datasource.pool.minSize">15</param>
    <param name="datasource.pool.maxSize">200</param>
    <param name="datasource.pool.maxWait">600000</param>
    <param name="datasource.pool.validationQuery">select 1 from dual</param>
    <param
name="datasource.connectionProperties">implicitCachingEnabled=true;maxStatements=100</param>
    <param name="datasource.pool.poolPreparedStatements">true</param>
    <param name="datasource.pool.maxOpenPreparedStatements">500</param>

    <param name="ks.core.datasource.url">${datasource.url}</param>
    <param name="ks.core.datasource.username">KSAPP</param>
    <param name="ks.core.datasource.password">KSAPP</param>
    <param name="ks.core.datasource.minSize">15</param>
    <param name="ks.core.datasource.maxSize">200</param>
    <param name="ks.core.datasource.maxWait">600000</param>
    <param name="ks.core.datasource.maxOpenPreparedStatements">500</param>

    <param name="keystore.file">/home/rice/rice.keystore</param>

    <!-- log4j settings -->
    <param name="log4j.settings.path">classpath: META-INF/log4j.properties</param>
    <!-- LUM Application URL -->
    <param
name="lum.application.url">http://appserv-2.ks.kuali.net/ks-with-rice-embedded</param>

    <!-- Tuning -->
    <param name="kim.cache.permission.max.size">3000</param>
    <param name="kim.cache.permission.max.age.seconds">90</param>
</config>
```

### **ks-with-rice-embedded-config.xml example**

```
<config>
    <!-- Please fill in proper values for these 2 parameters! -->
    <param name="appserver.url">http://appserv-2.ks.kuali.net</param>
    <param name="app.context.name">ks-with-rice-embedded</param>

    <param name="log4j.settings.path">classpath: log4j.properties</param>
    <param name="datasource.driver.name">oracle.jdbc.OracleDriver</param>

    <param name="datasource.url">jdbc:oracle:thin:@dbaddress:1521:SID</param>
```

```

<param name="datasource.username">KSRICE</param>
<param name="datasource.password">KSRICE</param>
<param name="datasource.pool.minSize">15</param>
<param name="datasource.pool.maxSize">100</param>
<param name="datasource.pool.maxWait">600000</param>
<param
name="datasource.connectionProperties">implicitCachingEnabled=true;maxStatements=100</param>
<param name="datasource.pool.poolPreparedStatements">true</param>
<param name="datasource.pool.maxOpenPreparedStatements">500</param>

<param name="ks.lum.datasource.url">${datasource.url}</param>
<param name="ks.lum.datasource.username">KSAPP</param>
<param name="ks.lum.datasource.password">KSAPP</param>
<param name="ks.lum.datasource.minSize">15</param>
<param name="ks.lum.datasource.maxSize">100</param>
<param name="ks.lum.datasource.maxWait">600000</param>
<param
name="ks.lum.datasource.connectionProperties">implicitCachingEnabled=true;maxStatement
s=100</param>
<param name="ks.lum.datasource.poolPreparedStatements">true</param>
<param name="ks.lum.datasource.maxOpenPreparedStatements">500</param>

<param name="ks.core.datasource.url">${datasource.url}</param>
<param name="ks.core.datasource.username">KSAPP</param>
<param name="ks.core.datasource.password">KSAPP</param>
<param name="ks.core.datasource.minSize">15</param>
<param name="ks.core.datasource.maxSize">100</param>
<param name="ks.core.datasource.maxWait">600000</param>
<param
name="ks.core.datasource.connectionProperties">implicitCachingEnabled=true;maxStatemen
ts=100</param>
<param name="ks.core.datasource.poolPreparedStatements">true</param>
<param name="ks.core.datasource.maxOpenPreparedStatements">500</param>

<param name="keystore.file">/home/kuali/rice.keystore</param>

<param name="ks.rice.url">http://appserv-1.ks.kuali.net/ks-rice-standalone</param>

<param name="serviceServletUrl">${application.url}/services/</param>

<!-- Rice URLs -->
<param
name="ks.rice.personLookup.serviceAddress">${ks.rice.url}/kr/lookup.do</param>
<param
name="ks.rice.actionList.serviceAddress">${ks.rice.url}/kew/ActionList.do</param>
<param
name="ks.rice.docSearch.serviceAddress">${ks.rice.url}/kew/DocumentSearch.do</param>
<param name="ks.ignore.rice.login">true</param>

```

```
</config>
```

### (CM 2.0) 3.2.3 Supported Properties for ks-rice-standalone

Table 4 lists the properties that are required as well as some of the most common optional config keys for the `ks-rice-standalone` properties file.

Property	Description	Use
appserver.url	This is the url for the root of the appserver (does not include the context name). ex: <code>http://my.school.edu:1234</code>	required
app.context.name	The name of deployed context	required
keystore.file	The file full path to the rice keystore file, currently located in the WAR WEB-INF/classes/rice.keystore	required
datasource.odb.platform	The DBMS that odb will use to map data	optional
datasource.driver.name	The java class signature for the JDBC driver. Default (oracle.jdbc.OracleDriver)	optional
datasource.url	The jdbc connection string for the rice datasource	required
datasource.username	username for rice datasource	required
datasource.password	password for rice datasource	required
datasource.btm.transactionTimeout	Transactional Connection timeout in seconds. Recommendation for load testing: <param override="false">120</param>	optional
datasource.btm.journal	Configure Bitronix logging. To disable Bitronix logging: <param override="false">null</param>	optional
datasource.pool.minSize	Initial and minimum size of open connections in the connection pool.	optional
datasource.pool.maxSize	Maximum size of open connections in the connection pool (increase when you start getting failed to establish connection errors)	optional
datasource.pool.maxWait	Maximum time to wait for a transaction to complete before timeout.	optional
ks.core.datasource.url	jdbc connection string for the core datasource	required
ks.core.datasource.username	core datasource username	required
ks.core.datasource.password	core datasource password	required
ks.core.datasource.minSize	Set the minimum db pool size for the core datasource.	optional
ks.core.datasource.maxSize	Set the maximum db pool size for the core datasource.	optional
ks.core.datasource.maxWait	Maximum time to wait for a transaction to complete before timeout.	optional
lum.application.url	The url for the ks-with-rice-embedded application including context. ex: <code>http://my.school.edu:1234/ks-with-rice-embedded</code>	required

ks.rice.url	The url for the ks-rice application including context. ex: <a href="http://my.school.edu:1234/ks-rice">http://my.school.edu:1234/ks-rice</a>	not used
log4j.settings.path	This changes the log4j config file that is used, should include the filename.	optional
kim.cache.permission.max.size	Number of cache records for kim permission cache	optional
kim.cache.permission.max.age.seconds	ttl for permission cache records	optional
rice.cxf.client.connectionTimeout	Timeout for service endpoint resolution (increase if necessary)	optional
rice.cxf.client.receiveTimeout	Timeout for service method response (increase if necessary)	optional
rice.cxf.client.allowChunking	Avoids message length computations and sends large responses in chunks (size is probably also controllable).	optional
additional.config.locations	Define location of additional configuration parameter files (use a system parameter)	optional

**Table 4: Supported properties for ks-rice-standalone**

#### (CM 2.0) 3.2.4 Supported Properties for ks-with-rice-embedded

Property	Description	Use
appserver.url	This is the url for the root of the appserver (does not include the context name). ex: [http://my.school.edu:1234]	required
app.context.name	The name of deployed context	required
keystore.file	The file full path to the rice keystore file, currently located in the WAR WEB-INF/classes/rice.keystore	required
datasource.odb.platform	The DBMS that odb will use to map data	optional
datasource.driver.name	The java class signature for the JDBC driver. Default (oracle.jdbc.OracleDriver)	optional
datasource.url	The jdbc connection string for the rice datasource	required
datasource.username	username for rice datasource	required
datasource.password	password for rice datasource	required
datasource.btm.transactionTimeout	Transactional Connection timeout in seconds. Recommendation for load testing: <param override="false">120</param>	optional
datasource.btm.journal	Configure Bitronix logging. To disable Bitronix logging: <param override="false">null</param>	optional
datasource.pool.minSize	Initial and minimum size of open connections in the connection pool.	optional
datasource.pool.maxSize	Maximum size of open connections in the connection pool (increase when you start getting failed to establish connection errors)	optional
datasource.pool.maxWait	Maximum time to wait for a transaction to complete before timeout.	optional
ks.lum.datasource.url	jdbc connection string for the lum datasource	required
ks.lum.datasource.username	lum datasource username	required

ks.lum.datasource.password	lum datasource password	required
ks.lum.datasource.minSize	Set the minimum db pool size for the lum datasource.	optional
ks.lum.datasource.maxSize	Set the maximum db pool size for the lum datasource.	optional
ks.lum.datasource.maxWait	Maximum time to wait for a transaction to complete before timeout.	optional
ks.core.datasource.url	jdbc connection string for the core datasource	required
ks.core.datasource.username	core datasource username	required
ks.core.datasource.password	core datasource password	required
ks.core.datasource.minSize	Set the minimum db pool size for the core datasource.	optional
ks.core.datasource.maxSize	Set the maximum db pool size for the core datasource.	optional
ks.core.datasource.maxWait	Maximum time to wait for a transaction to complete before timeout.	optional
lum.application.url	The url for the ks-with-rice-embedded application including context. ex: [http://my.school.edu:1234/ks-with-rice-embedded]	not used
ks.rice.url	The url for the ks-rice application including context. ex: [http://my.school.edu:1234/ks-rice]	required
ks.rice.personLookup.serviceAddress	<param>\${ks.rice.url}/kr/lookup.do</param>	required
ks.rice.actionList.serviceAddress	<param>\${ks.rice.url}/kew/ActionList.do</param>	required
ks.rice.docSearch.serviceAddress	<param>\${ks.rice.url}/kew/DocumentSearch.do</param>	required
ks.ignore.rice.login	Set to *true* if not using a single sign on for both RICE and KS	required
log4j.settings.path	This changes the log4j config file that is used, should include the filename. The default log4j settings are in the project <a href="#">here</a>	optional
ks.core.jpa.showSql	Set hibernate statement logging for the core datasource (true or false).	optional
ks.lum.jpa.showSql	Set hibernate statement logging for the lum datasource (true or false).	optional
rice.cxf.client.connectionTimeout	Timeout for service endpoint resolution (increase if necessary)	optional
rice.cxf.client.recieveTimeout	Timeout for service method response (increase if necessary)	optional
rice.cxf.client.allowChunking	Avoids message length computations and sends large responses in chunks (size is probably also controllable).	optional
additional.config.locations	define location of additional configuration parameter files (use a system parameter)	optional

**Table 3: Supported properties for ks-with-rice-embedded**

## (CM 2.0) 3.3 Deploying the Application in Tomcat

After you have downloaded the WAR files and defined the properties it is now time to deploy the application in Tomcat.

**i** When running in the distributed configuration (2 servlet containers), it is advisable to start the servlet container with the `ks-rice-stanalone-xxx.war` first. Once that application is fully initialized and operational, you can proceed with starting the servlet container with the `ks-with-rice-embedded-xxx.war`.

To deploy the application in Tomcat perform the following steps (if you are not using Tomcat as the application server, you will need to perform appropriate steps pertinent to that server).

1. Add the following memory settings to your server configuration

```
-Xmx1024m -XX:PermSize=256m
```

*For Unix, add these settings to CATALINA\_OPTS environment variable for the user starting tomcat.*

2. Add the Oracle JAR to your Tomcat server's classpath (typically done by copying the JAR in `TOMCAT_HOME/lib` directory).
3. Copy the WAR files into `TOMCAT_HOME/webapps` (it is recommended that the version number be removed from the war filename first).
4. Start Tomcat

## (CM 2.0) 3.4 Accessing the Application

To access the application, go to the `appserv.url` that you specified in your property file.

You should be presented with the login screen. By default, the administrative user name is "admin" and the password is "admin".

**Notes:** An index page will be available at [<http://yoururl.edu/warname/>] where `_warname_` is the name of the WAR file you deployed.

To find other predefined users, go into Rice, click the Administration tab, then click the person link to search for usernames. The default password for all users is the username.

## (CM 2.0) 4. Configuring KS Curriculum Management

*Configuration* is altering KS Curriculum Management in prescribed methods to meet an institution's business requirements and objectives. When these prescribed methods are used, implementers can manage configurations independently from the core product and the institutional changes will be preserved throughout subsequent releases of KS Curriculum Management.

Configuring KS Curriculum Management involves activities such as changing CSS (Cascading Style Sheet) files, XML (Extensible Markup Language) files, and Java classes. For services, configuration involves replacing an entire service with an institutional version.

Table 5 illustrates the parts of the system that can be configured at each level of the architecture---the user interface, the application and the services.

Architectural Layer	What is Configured	Configuration Tool
User Interface	Theme---fonts, colors, images, etc.	Cascading Style Sheet (CSS)
	Labels and messages	KS Admin Tool (Kuali RICE)
	View layout, additional views, adding/dropping fields	Course Configurer, GWT Deffered Binding
Curriculum Development Application	Adding fields using dynamic attributes	Service Dictionaries
	Updating search, lookups and enumerations	Search/lookup Config XMI and KS Admin Tool for enumeration
	Business workflow---adding users, adding/changing steps and responsibilities	Kuali Enterprise Workflow (KEW)
Services	Types, States and Data Constraints	Service Dictionaries

**Table 5: What can be configured in KS Curriculum Management**

## (CM 2.0) 4.1 Other Deployment Considerations

When deploying KS Curriculum Management in your existing environment, there are other things you might want to consider, such as the service bus or the extraction of data from the system for use in existing systems.

## **(CM 2.0) 4.1.1 The Service Bus**

Kuali Student uses the Kuali Service Bus (KSB). A part of the KSB will be embedded in the KS application to provide service location and reliability as well as support for synchronous and asynchronous calls. In this configuration, there will be no support for external web services not deployed on a KSB instance, nor will there be message translation, complex routing patterns or native legacy support.

Implementing institutions will need to consider how KS's use of KSB fits into their larger Enterprise Integration Pattern (EIP) strategy. Approaches an institution could take include:

- Deploying KSB as a client of another service bus
- Deploying a web service client proxy on KSB
- Write java objects that expose legacy systems to the bus
- Write services (xslt or java ) that apply routing logic to a service call.



## **KS Curriculum Management 2.0 Configuration Guide**

(CM 2.0) 1. Configuration Overview .....	4
(CM 2.0) 1.1 What is Configuration? .....	4
(CM 2.0) 1.1.1 What Can be Configured in KS Curriculum Management 2.0? .....	4
(CM 2.0) 1.1.2 Configuration vs. Customization .....	5
(CM 2.0) 1.2 What Skill Sets are Required for Configuration? .....	5
(CM 2.0) 1.2.1 User Interface .....	5
(CM 2.0) 1.2.2 Configuring Searches and Enumerations .....	6
(CM 2.0) 1.2.2.1 Service and Data Configuration .....	6
(CM 2.0) 1.2.2.2 Business Processes and Workflow .....	6
(CM 2.0) 1.2.2.3 Authentication .....	6
(CM 2.0) 1.2.2.4 Deployment Topology .....	7
(CM 2.0) 2. Setting up an Institution Configuration Project .....	7
(CM 2.0) 2.1 How to Set up the Configuration Project .....	7
(CM 2.0) 3. Configuring the User Interface .....	8
(CM 2.0) 3.1 Elements of the User Interface .....	8
(CM 2.0) 3.2 Configuring the Theme .....	10
(CM 2.0) 3.2.1 Implementing Your Own Theme .....	10
(CM 2.0) 3.2.2 Changing CSS .....	10
(CM 2.0) 3.2.3 Replacing CSS .....	10
(CM 2.0) 3.2.4 Replacing Images .....	11
(CM 2.0) 3.3 Configuring the Layout of Views, Sections, and Fields .....	11
(CM 2.0) 3.3.1 Extending CourseConfigurer .....	11
(CM 2.0) 3.3.2 Defining Institutional Classes .....	12
(CM 2.0) 3.3.3 Adding to and Changing a Layout's Sections and Navigation .....	12
(CM 2.0) 3.3.4 Changing the Section Order in a View or Section .....	12
(CM 2.0) 3.3.5 Removing a Section .....	13
(CM 2.0) 3.3.6 Adding a Section .....	13
(CM 2.0) 3.4 Configuring Fields within the User Interface .....	14
(CM 2.0) 3.5 Configuring Labels, Messages, and Help Text .....	15
(CM 2.0) 4. Configuring the Service Dictionaries .....	15
(CM 2.0) 4.1 Dictionary Schema .....	15
(CM 2.0) 4.1.1 The Spring Bean Schema .....	16
(CM 2.0) 4.1.2 Parent-Bean Pattern .....	16
(CM 2.0) 4.2 Configuring Constraints .....	16
(CM 2.0) 4.2.1 Constraint Properties .....	16
(CM 2.0) 4.2.2 Base Template Beans .....	19
(CM 2.0) 4.2.3 Writing Custom Validators .....	19
(CM 2.0) 4.2.4 Overriding KS Configurations .....	20
(CM 2.0) 4.3 Adding a Field (Dynamic Attributes) .....	21
(CM 2.0) 5. Configuring Searches, Pickers, and Browses .....	22
(CM 2.0) 5.1 Before You Begin .....	23
(CM 2.0) 5.2 Configuring Search Criteria and Results .....	23
(CM 2.0) 5.2.1 Example of Adding a New Criteria Parameters to the Search Type .....	23
(CM 2.0) 5.2.2 Cross Service Searches .....	24
(CM 2.0) 5.3 Field Definitions .....	24
(CM 2.0) 5.4 Lookup Definitions and Lookup Field Bindings .....	25
(CM 2.0) 5.4.1 Constraints .....	26
(CM 2.0) 5.4.2 Lookup Parameters .....	27
(CM 2.0) 5.4.3 Lookup Results .....	28
(CM 2.0) 5.5 Field Binding .....	28
(CM 2.0) 5.5.1 Type Aware Field Binding .....	28
(CM 2.0) 5.5.2 Configuring a Search-Select Widget for a Lookup .....	29
(CM 2.0) 5.6 Configuring Enumerations .....	29
(CM 2.0) 5.6.1 Adding or Changing Enumeration Value Sets .....	29
(CM 2.0) 5.6.2 Configuring a Widget to Use an Enumeration .....	29
(CM 2.0) 5.7 Configuring the Catalog Browser .....	29
(CM 2.0) 5.7.1 Displaying the Catalog Browser in the User Interface .....	29
(CM 2.0) 5.7.1.1 Searches and Lookups for Browse by Subject Area .....	30
(CM 2.0) 5.7.1.2 Searches and Lookups for Browse by School or College .....	30
(CM 2.0) 5.7.2 Configuring What is Displayed on the Screen .....	30
(CM 2.0) 5.7.2.1 Hide or Show a Column .....	30
(CM 2.0) 5.7.2.2 Change the Column Heading .....	31
(CM 2.0) 5.7.2.3 Change the Column Order .....	31
(CM 2.0) 5.8 Configuring the Program Browser .....	31
(CM 2.0) 5.8.1 Configuring Facets for Browse Filter .....	31
(CM 2.0) 6. Configuring Business Processes and Workflow .....	34
(CM 2.0) 6.1 What is NOT Covered in this Document .....	34
(CM 2.0) 6.2 Kuali Workflow Documents and KS Curriculum Management Proposals .....	34
(CM 2.0) 6.2.1 Configuring Workflow Document Content and Document Title .....	35
(CM 2.0) 6.2.2 Configuring Fields Required by Workflow Node .....	35
(CM 2.0) 6.3 Document Types .....	38
(CM 2.0) 6.4 Route Nodes .....	39

(CM 2.0) 6.4.1 Organization Routing Node Configurations .....	40
(CM 2.0) 6.4.2 Qualifier Resolver Classes .....	40
(CM 2.0) 6.4.2.1 Committee on Curricula Organization Type node Configuration (Org Tree) .....	40
(CM 2.0) 6.4.2.2 Static Organization Node Configuration .....	41
(CM 2.0) 6.4.2.3 Committee on Curricula Organization Node Configuration (Specific Org) .....	42
(CM 2.0) 6.4.3 Workflow Post Processing Configuration .....	43
(CM 2.0) 6.4.3.1 KS Curriculum Management Post Processor Base .....	43
(CM 2.0) 6.4.3.2 KS Curriculum Management Course Post Processor Base .....	44
(CM 2.0) 6.4.4 Kuali Identity Management Constructs Used by Workflow .....	44
(CM 2.0) 6.4.4.1 Principals .....	44
(CM 2.0) 6.4.4.2 Roles .....	44
(CM 2.0) 6.4.4.3 Responsibilities .....	45
(CM 2.0) 7. Configuring Rules .....	45
(CM 2.0) 8. Authorization .....	51
(CM 2.0) 8.1 Field Level Authorization .....	51
(CM 2.0) 8.1.1 Restricting Field Access .....	51
(CM 2.0) 8.1.1.1 Adding a Field Restriction .....	52
(CM 2.0) 8.1.2 Defining Field Permissions .....	52
(CM 2.0) 8.1.3 Grant Field Permission .....	54
(CM 2.0) 8.2 KS Admin Screen Permissions and Role .....	54
(CM 2.0) 8.3 Creating Derived Role .....	55
(CM 2.0) 9. Authentication .....	57
(CM 2.0) 9.1 Spring Security .....	57
(CM 2.0) 9.1.1 Overrides to Support Kuali Identity Management (KIM) .....	57
(CM 2.0) 9.2 Integrating Other Authentication Systems with KS Curriculum Management .....	57
(CM 2.0) 9.3 Authentication with KS Curriculum Management and Rice .....	58
(CM 2.0) 9.4 Configuring CAS with KS Curriculum Management Embedded .....	58
(CM 2.0) 10. Configuring Type Data .....	59
(CM 2.0) 10.1 Degree Types .....	59
(CM 2.0) 11. Misc Configuration .....	59
(CM 2.0) 11.1 Copy Course & Copy Course Proposal - Fields to Copy .....	59

# KS Curriculum Management 2.0 Configuration Guide

## (CM 2.0) 1. Configuration Overview

This document provides details on how to configure various aspects of KS Curriculum Management 2.0, including the user interface, data, business rules and processes, enumerations and searches.

This document provides information necessary for the following decision-makers to make informed implementation decisions:

- **Business Analysts** on implementation teams who need to translate the business requirements of Subject Matter Experts (SME's) into configuration requirements.
- **Lead developers** who will be working closely with business analysts to configure KS Curriculum Management to meet institutional requirement

### (CM 2.0) 1.1 What is Configuration?

Configuration is altering KS Curriculum Management in prescribed methods to meet an institution's business requirements and objectives. When these prescribed methods are used, implementers can manage configurations independently from the core product and the institutional changes will be preserved throughout subsequent releases of KS Curriculum Management.

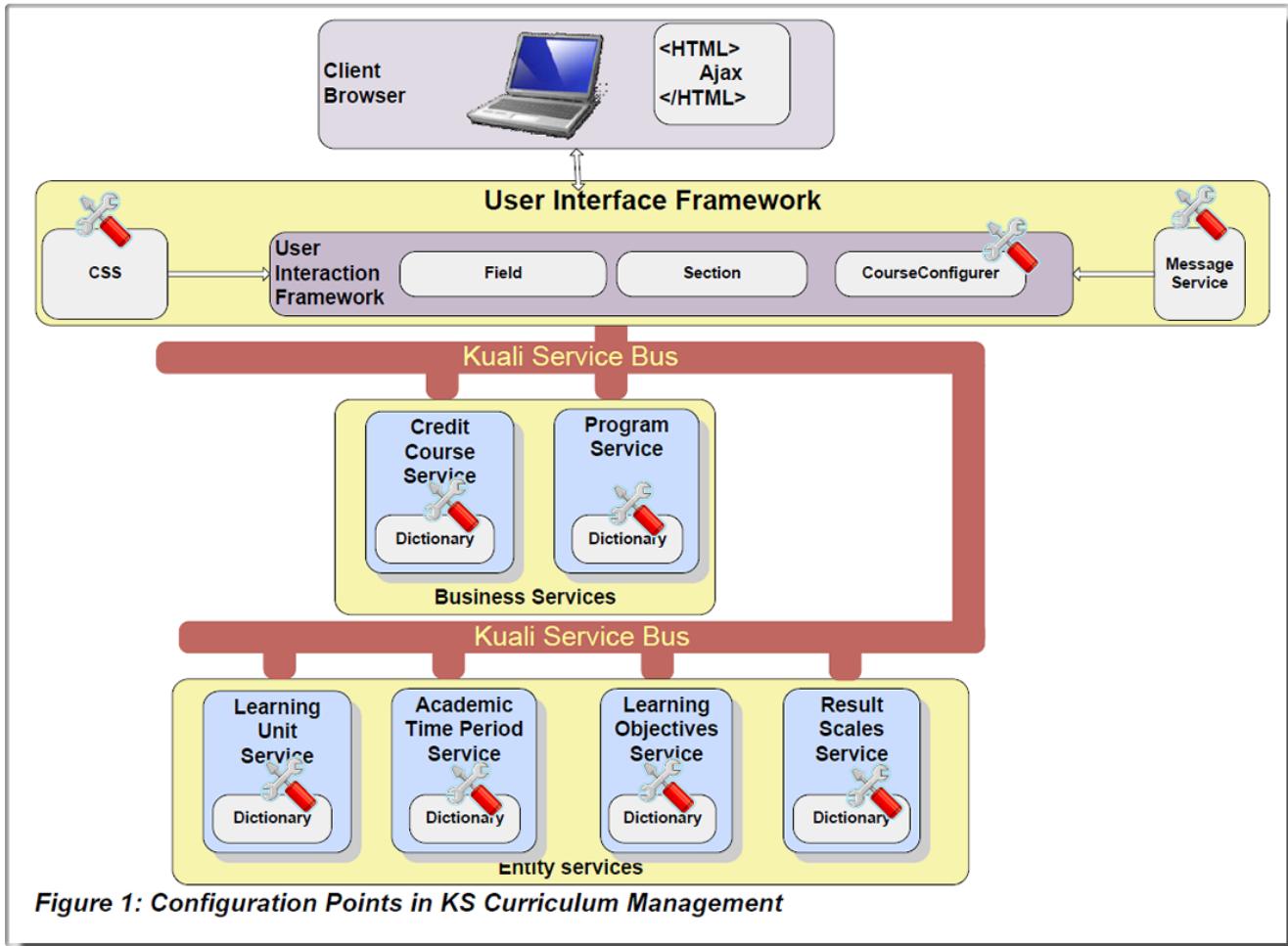
Configuring KS Curriculum Management involves activities such as changing CSS (Cascading Style Sheet) files, XML (Extensible Markup Language) files, and Java classes. For services, configuration involves replacing an entire service with an institutional version.

#### (CM 2.0) 1.1.1 What Can be Configured in KS Curriculum Management 2.0?

The table below illustrates the parts of the system that can be configured at each level of the architecture---the user interface, the application and the services.

Architectural Layer	What is Configured	Configuration Tool
User Interface	Theme-fonts, colors, images, etc.	Cascading Style Sheet (CSS)
	Labels and messages	KS Admin Tool (Kuali Rice)
	View layout, additional views, adding or removing fields	Course Configurer, GWT Deferred Binding
Curriculum Development Application	Adding fields using dynamic attributes	Service Dictionaries
	Updating search, lookups and enumerations	Search/lookup Config XMI and KS Admi Tool for enumeration
	Business workflow - adding users, adding/changing steps and responsibilities	Kuali Enterprise Workflow (KEW)
Services	Types, States and Data Constraints	Service Dictionaries

Figure 1 further illustrates these mechanisms as indicated by the tools ( ) symbol in the diagram.



**Figure 1: Configuration Points in KS Curriculum Management**

## (CM 2.0) 1.1.2 Configuration vs. Customization

Customization is changing the "core product" of Kuali Student so that changes will not be preserved throughout subsequent releases. The core product of Kuali Student includes:

- A set of Service Contracts. Services are divided into two categories:
  - Core entity services, such as the Learning Unit service
  - Business services such as the CreditCourse service
- Mechanisms for constraining the data in the service message structures
- A mechanism for translating service message data into a meta-data structure for consumption by the User Interface (to support authorization and UI validation)
- A user interface framework
  - ks-common-ui

These artifacts should **not** be changed independently by implementing institutions. Institutions that identify necessary changes to the core product should follow established community processes for suggesting changes.

## (CM 2.0) 1.2 What Skill Sets are Required for Configuration?

The following tables in this section list the various aspects of KS Curriculum Management that can be configured, a summary of how that aspect is configured and a list of skill sets required.

A relative indicator of the level of complexity or difficulty involved in making the change is also included for each configuration. These three levels include:

- **Level 1:** The change is relatively straightforward and does not require or affect other configurable aspects of the system.
- **Level 2:** The change requires some up-front design and coordination between people with different skill sets (usually Business Analysts and Developers) and/or more than one aspect of the system (User Interface, Dictionaries, etc.) need to be configured simultaneously to achieve the desired result.
- **Level 3:** Similar to Level 2, but requires significant up-front design and coordination or requires specific knowledge of systems external to Kuali Student (for example, Authentication systems)

## (CM 2.0) 1.2.1 User Interface

Aspects of the user interface that can be configured include the overall theme, view layouts and labels and messages displayed to the user.

What	How	Who (skill Sets)	Difficulty
Theme (general look and feel, graphics, fonts, etc.)	Cascading Style Sheet (CSS)	User Interface Designer Web Designer	Level 1 (Java Dev. required only for initial setup)
Headers and Navigation Elements	Deferred binding/GWT code	UI Designer Business Analyst Java/GWT developer	Level 2
Menu and Field in a Layout	Java/GWT code Extending CourseConfigurer	UI Designer Business Analyst Java/GWT developer	Level 2
Appearance of a Field	Deferred binding/GWT code	UI Designer Business Analyst Java/GWT developer	Level 2
Labels, Messages and Help Text	KS Admin Client	Business Analyst	Level 1

## (CM 2.0) 1.2.2 Configuring Searches and Enumerations

Named searches use a set of criteria to search for data that may be maintained across multiple services of record, while enumerations are lists of valid values for a particular data element (e.g., department codes, country codes, etc.).

What	How	Who (skill sets)	Difficulty
Search	Search.xml (Spring bean override pattern)	BA Developer	Level 2
Enumerations	KS Admin Client and Search.xml (Spring bean override pattern)	BA Developer	Level 2

### (CM 2.0) 1.2.2.1 Service and Data Configuration

Services and data are configured through the corresponding dictionaries.

What	How	Who (Skill Sets)	Difficulty
Service Dictionaries	Dictionary XML (Spring bean override pattern)	BA Java developer	Level 2

### (CM 2.0) 1.2.2.2 Business Processes and Workflow

Business processes and workflow are administered through KIM (Kuali Identity Management) and KEW (Kuali Enterprise Workflow). This document explains how these are used in KS Curriculum Management---details for how to administer workflow and identities are provided in the Kuali Rice documentation.

What	How	Who (Skill Sets)	Difficulty
Changing a route pattern	KEW (Kuali Enterprise Workflow) document type xml and KIM (Kuali Identity Management) data	BA Java Developer	Level 2
Adding users to an existing routing pattern	Roles defined in KIM (Kuali Identity Management)	BA Subject matter expert	Level 2
Authorization	Create permissions and roles in KIM (Kuali Identity Management)	BA	Level 2

### (CM 2.0) 1.2.2.3 Authentication

Authentication can use local directory services or can be integrated with an existing Single Sign-On infrastructure.

What	How	Who (Skill Sets)	Difficulty
Authentication	Point KS to institutional repository (LDAP, Active Directory)	Developer	Level 2
	Replace with single sign-on	Developer	Level 2
	Replace AOP wrapper	Developer	Level 3

#### (CM 2.0) 1.2.2.4 Deployment Topology

Changing how KS Curriculum Management is deployed requires very specific technical skills.

What	How	Who (Skill Sets)	Difficulty
Deployment topology	Maven Projects	Deployment Managers Java developers	Level 3

## (CM 2.0) 2. Setting up an Institution Configuration Project

To ensure that your institution's configurations are maintained through future releases, you will want to set up an institution configuration project. You can maintain all changes to the GWT and Java code regardless of the project to which it relates (common, core, lum, etc.). You will use this project to build a war file with your institution's specific configuration code.

Since the institution-specific code in the configuration project is compiled to the classes folder in the war file, the classes and resources have classpath precedence over any classes and resources with the same names in dependent libraries such as ks-core, ks-lum, etc. This allows you to redefine library classes for bug fixes or additional functionality in situations where KS Curriculum Management cannot be changed.

### (CM 2.0) 2.1 How to Set up the Configuration Project

Follow this procedure to set up the bundled institution configuration project.

 The easiest way to get started is to export the example configuration project, [ks-sample-config-cm-2.0](#), and rename it to match your school. This project is in its infancy and as such may not reflect best practices of implementation and configuration but is a starting point for implementers to review as part of their initial stages of development. There are cases (which we are looking to correct) where it may be necessary to replicate an entire module (e.g. ks-lum-ui) to correct bugs. It is NOT recommended that implementers replicate the entire code base as this will make upgrades much more complicated. In most cases if implementers need to make changes to the code base it is likely due to unknown bugs, or lack of configuration points, each of which should be easily incorporated into the next patch release via a [Bug Fix Contribution](#).

To create this sample config the following procedure was followed:

1. Exported ks-with-rice-bundled as ks-sample-config-project-cm-2.0

 It should be possible to do the same thing for embedded deployments with the ks-with-rice-embedded branch, currently at <https://svn.kuali.org/repos/student/enrollment/ks-deployments/tags/ks-deployments-2.0.1-cm/ks-web/ks-with-rice-embedded/>.

2. Layered custom configurations on top:

- a. SampleLuMain.gwt.xml module definition
- b. Example java overrides:
  - i. SampleCourseAdminConfigurer.java
  - ii. SampleCourseAdminWithoutVersionConfigurer
  - iii. SampleClientBundle.java
  - iv. SampleCssImpl.java
- c. Example Resources
  - i. ks-sample-override-config.xml
  - ii. ks-sample-courseInfo-dictionary-context.xml
  - iii. SampleOverrides.css
- d. Example SQL update statements to the messages
  - i. sql/Update Messages.sql that updates the messages for the KSHeader
  - ii. sql/Display Messages.sql that displays the messages

3. Changed the pom.xml

- a. Renamed the project to match
  - i. artifact id
  - ii. name
  - iii. description
- b. Tell GWT to compile the SampleLuMain gwt module instead of the LuMain one

```
<ks.gwt.module.lum>edu.kuali.config.lum.lu.ui.SampleLUMMain</ks.gwt.module.lum>
```

4. Changed META-INF/ks-with-rice-bundled-config.xml

- a. ⚠ Don't get confused and edit ks-with-rice-bundled-**context**.xml!
- b. Added a line to point to the dictionary overrides

```
<!--include the override-->
<param
name="config.location">classpath:ks-sample-override-config.xml</param>
```

- c. Note how ks-sample-override-config.xml overrides the ks.lum.dictionary.serviceContextLocations by appending the ks-sample-courseInfo-dictionary-context.xml at the end.

## (CM 2.0) 3. Configuring the User Interface

The KS Curriculum Management interface can be configured on a number of different levels, including the general look and feel, labels and messages, and view layouts. Each of these levels has its own configuration mechanisms and methods. The following table summarizes how these different aspects of the user interface are configured and what skill set(s) are required.

What	How	Who (skill Sets)	Difficulty
Theme (general look and feel, graphics, fonts, etc.)	Cascading Style Sheet (CSS)	User Interface Designer Web Designer	Level 1 (Java Dev. required only for initial setup)
Headers and Navigation Elements	Deferred binding/GWT code	UI Designer Business Analyst Java/GWT developer	Level 2
Menu and Field in a Layout	Java/GWT code Extending CourseConfigurer	UI Designer Business Analyst Java/GWT developer	Level 2
Appearance of a Field	Deferred binding/GWT code	UI Designer Business Analyst Java/GWT developer	Level 2
Labels, Messages and Help Text	Message Utility in Rice if a message key exists	Business Analyst	Level 1

### (CM 2.0) 3.1 Elements of the User Interface

Figure 2 illustrates the elements of the KS Curriculum Management interface.

**Kuali Student**

Select an area... ▾

Home » Curriculum Management » New Course (Proposal) » Course Information

## New Course (Proposal)

Proposal Status: Draft | [Comments](#) | [Decisions](#)

**COURSE SECTIONS**

- Course Information** [print]
- Governance
- Course Logistics
- Learning Objectives
- Course Requisites
- Active Dates
- Financials
- Authors & Collaborators
- Supporting Documents
- [Review Proposal](#)

### Course Information

**Proposal Title\*** [?]

This title is used for identifying the proposal through the approval process.  
It is highly recommended that you use the same name for the course and proposal title.

**Course Title\***

It is highly recommended that you use the same name for the course and proposal title.

**Transcript Course Title**

This is the truncated title that will appear on the printed transcript.

24 characters left

**Subject Code** Required on Submit Course Number ?

Must be 4 letter code      Must be 3 digit number

[Cross List, Offer Jointly, Add Version Codes ►](#)

**Instructor(s)** [?]

Add to list

[Advanced Search](#)

**Description and Rationale**

**Description** Required on Submit

This description will appear in the catalog.

**Proposal Rationale** Required on Submit

This will be used in evaluating the proposal. It will not be published in the catalog.

**Figure 2: Elements of the User Interface**

The following are definitions of the elements shown in Figure 2:

- **LayoutController:** A LayoutController is both a view and a controller. A LayoutController defines the setup and layout of its child views. The view of the LayoutController will change based on what views are added to it.

For example, the Course Proposal LayoutController includes the views Governance, Course Format, Authors and Collaborators, etc. Navigation links are automatically added to the layout and handled by their LayoutController. The LayoutController also manages the saving and receiving of the model used by its child sections.

In Figure 2, the LayoutController presents the entire Course Proposal, including what is displayed in the browser currently (Course Information) as well as what is available through other views (Governance, Course Logistics, Learning Objectives, etc.) as the user navigates them.

- **SectionView:** A SectionView is a meaningful group of the subsections and fields within a View. A SectionView is defined for presentation purposes only, and has no mapping to the corresponding service. The SectionView typically corresponds to a single navigation item. Changes made to a section need to be saved before the user navigates to another section or the changes will be lost. There may be any

number (reasonable for layout in the UI) of SectionViews within a LayoutController.

In Figure 2, the current SectionView is Course Information, and shows subsections and fields related to the logistics of the proposed course.

- **VerticalSection:** A VerticalSection is a layout of fields and/or other sections in a vertical arrangement. A VerticalSection may be nested within a SectionView or another VerticalSection.

In Figure 2, the entire SectionView is defined as a VerticalSection.

- **GroupSection:** A GroupSection defines one or more lines of fields. You can define where lines (rows) of fields break to allow fields to be displayed next to each other. Any sections added to a GroupSection appear below the grouped fields, not inline as in a VerticalSection.

In Figure 2, the Subject Code and Course Number fields are defined within a GroupSection.

Field: A field is a widget which is bound to an application data value and to its validation (for example, the name of the person proposing a course). Fields are defined by the FieldDescriptor class and contain not only the input field but also its label, instructions, and constraint text if there is any defined.

## (CM 2.0) 3.2 Configuring the Theme

The overall "look and feel" of the user interface, or theme, includes aspects such as graphics, fonts, and colors. In KS Curriculum Management, these are managed through Cascading Style Sheets (CSS). By changing or replacing the CSS and any corresponding image files, you can change the interface to display your institution's logo, school colors and other aspects to closely match the institutional website.

Styles selectors are specified in CSS in the Common and LUM project bundles. The entry points for Common and LUM injects all CSS resources that are also found in the bundles directly into the JSP and HTML file that are served. The result is that a single file is downloaded for the HTML and CSS in a single call to the server.

### (CM 2.0) 3.2.1 Implementing Your Own Theme

To change aspects of the default theme with your own, follow these steps:

1. Create a gwt module to contain the theme classes and CSS to override in your project.
  - a. Create a \*.gwt.xml file in an appropriately named directory for your theme. This step is not necessary if your theme is part of the parent gwt module. If it is not part of the parent gwt module you will need to modify the parent gwt module definition to inherit it.
  - b. Create a client folder in the theme directory.
  - c. Create a public folder in the theme directory and create css and images folders within the public folder
2. Create classes which will replace the theme classes defined in ks-common and ks-lum where appropriate.

The recommended way to replace classes is to extend the class you are intending to replace and change only the methods that need to be changed. These classes are located in \*.theme.standard.client packages in their respective projects.

### (CM 2.0) 3.2.2 Changing CSS

To change a style in the default CSS in your institutional project, follow these steps:

1. Determine the style name you are trying to replace by using a web development tool or inspecting the generated source.
2. Create a new CSS file in the css folder of your theme directory.
3. Define the override for your chosen selector in your css file.
4. Create a client bundle in the client directory, which extends the LumClientBundle, and follow the same format for adding a new css file as seen in LumClientBundle, but pointing it to the new css resource you created.
5. Create a class to override LumCssImpl that also extends it, and point it to an instance of your new bundle in its getCssString() method (copy the LumCssImpl as a starting point).
6. In your \*.gwt.xml file use GWT deferred binding to replace instances of LumCssImpl with your new implementation. For example:

```
<replace-with  
class="edu.kuali.config.lum.lu.ui.client.theme.SampleCssImpl">  
  <when-type-is  
  class="org.kuali.student.lum.ui.theme.standard.client.LumCssImpl"/>  
</replace-with>
```

After this initial setup, you can use the CSS file you created to add any additional CSS selectors you want to override.

### (CM 2.0) 3.2.3 Replacing CSS

To completely overwrite the CSS of a project with your own (for large changes):

1. Create your own theme implementation which extends *Theme* or *LumTheme*.
2. Create additional classes needed by your theme. Use the theme classes in the projects as a guide.
3. Put all your CSS files in the *css directory*. It is recommended that you copy the css files from the project for which you are replacing the

css and change it, and the corresponding *ClientBundle*, as needed.

4. In your \*.gwt.xml file use GWT deferred binding to replace instances of *Theme* or *LumTheme* with your new implementation. See *StandardTheme.gwt.xml* for an example.

### (CM 2.0) 3.2.4 Replacing Images

To replace a single image, such as the logo displayed on every view, follow these steps:

1. Copy your image file to the images folder for your theme.
2. Create a client bundle in the client directory, which extends the LumClientBundle (or KSClientBundle if you are replacing common images), and override the method which points to the image you want to replace.
3. Create a class to override the corresponding ImagesImpl class, which also extends it, and override the method that gets the image you want to replace.
4. In your \*.gwt.xml file use GWT deferred binding to replace instances of the ImagesImpl with your new implementation. See 3.2.2 Changing CSS on page 18 for an example of deferred binding.
5. You will need to recompile the application for the change to take effect. If the image exists as a background image in the CSS, use the Changing CSS method above.

### (CM 2.0) 3.3 Configuring the Layout of Views, Sections, and Fields

You can configure a view (for example, Course Proposal) by adding, removing or changing the order of sections (for example, Governance, Course Format, Authors and Collaborators) within that view. Likewise, you can configure a section by adding, removing or changing the order of sections and fields within that section.

The following files contain the layouts for the views in KS Curriculum Management.

- CurriculumHomeConfigurer
- CourseConfigurer
- ViewCourseConfigurer
- KS Curriculum Management 1.1 03/04/11 Page 20 of 68
- CourseSummaryConfigurer
- MajorEditConfigurer
- MajorViewConfigurer
- VariationEditConfigurer
- VariationViewConfigurer

#### (CM 2.0) 3.3.1 Extending CourseConfigurer

Before changing the layout of views, sections and fields, you need to create institutional extensions to these within your institutional configuration project. You will then make any changes to the layout in these files.

1. Create an institution configuration project as described in Chapter 2. Setting Up an Institution Configuration Project.
2. Create an institution CourseConfigurer (e.g., SampleCourseConfigurer.java) that extends the standard CourseConfigurer.
3. Create an institution GWT module file (e.g., SampleLUMMain.gwt.xml) within your institution configuration project.
4. Copy the entire contents of the standard GWT module file (LUMMain.gwt.xml) into the institution entry point.
5. Edit the institution GWT module file using deferred binding to replace the standard CourseConfigurer with your institution-specific CourseConfigurer as shown by the replace-with section below:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<module>
    <inherits name="org.kuali.student.lum.lu.ui.main.LumUI"/>
    <entry-point
        class="org.kuali.student.lum.lu.ui.main.client.LUMMainEntryPoint"/>
    <replace-with class="[domain].ui.configuration.SampleCourseConfigurer">
        <when-type-is
            class="org.kuali.student.lum.lu.ui.course.client.configuration.
CourseConfigurer"/>
    </replace-with>
</module>
```

6. Edit the your project's pom.xml file to use the new GWT module by commenting the GWT entry point and adding your institution-specific entry point.

```

7. <properties>

<ks.gwt.module.org>org.kuali.student.core.organization.ui.OrgEntry</ks.gwt.module.org>
    <!-- old GWT entry point for Curriculum Management

<ks.gwt.module.lum>org.kuali.student.lum.lu.ui.main.LUMMain</ks.gwt.module.lum>
    -->

<ks.gwt.module.lum>ca.ubc.student.kuali.lum.lu.ui.main.SampleLUMMain</ks.gwt.module.lum>
</properties>

```

## (CM 2.0) 3.3.2 Defining Institutional Classes

New classes and widgets need to be placed in the default folders of your institution GWT module or in a source folder defined in the module XML.

If your GWT module **XML** is:

{module XML package}/SampleLUMMain.gwt.xml

then the module default folders are:

- **source:** {module XML package}/client
- **images, css, etc:** {module XML package}/public

## (CM 2.0) 3.3.3 Adding to and Changing a Layout's Sections and Navigation

A LayoutController can contain a variety of views and ways to handle navigating to them. The LayoutController used for Course and Program is the MenuSectionController.

To add views to the menu on the left hand side of the layout, you must define a SectionView to add, then add it to the layout through the *addMenuItem(String parentMenuName, SectionView view)* call. The order in which the menu items are added determines the order in which they are displayed in the user interface.

**Note:** You can also add views through the *addView* call, but they are not automatically added to the left hand menu and must be navigated to through a custom widget using the *showView* call.

Below is an example of adding *SectionViews* to a menu in *CourseConfigurer*.

```

String sections = getLabel(LUConstants.COURSE_SECTIONS);
layout.addMenu(sections);

//Course Content
layout.addMenuItem(sections, generateCourseInfoSection());
layout.addMenuItem(sections, generateGovernanceSection());
layout.addMenuItem(sections, generateCourseLogisticsSection());

```

## (CM 2.0) 3.3.4 Changing the Section Order in a View or Section

Within a view or section, sections and fields are presented in the order in which they are specified in the configuration file in your institution configuration project.

To change the order of sections or fields, change the order of the *addSection* or *addField* statements in your configuration file. You will need to recompile the application for the changes to take effect.

Below is an example of a method defining the sections and fields within the Course Information view.

```

public SectionView generateCourseInfoSection() {
    VerticalSectionView section =
        initSectionView(CourseSections.COURSE_INFO, LUConstants.INFORMATION_LABEL_KEY);

    addField(section, PROPOSAL_TITLE_PATH,
generateMessageInfo(LUConstants.PROPOSAL_TITLE_LABEL_KEY));

    addField(section, COURSE + "/" + COURSE_TITLE,
generateMessageInfo(LUConstants.COURSE_TITLE_LABEL_KEY));

    addField(section, COURSE + "/" + TRANSCRIPT_TITLE,
generateMessageInfo(LUConstants.SHORT_TITLE_LABEL_KEY));

    section.addSection(generateCourseNumberSection());

    FieldDescriptor instructorsFd = addField(section, COURSE + "/" +
INSTRUCTORS, generateMessageInfo(LUConstants.INSTRUCTORS_LABEL_KEY));
    instructorsFd.setWidgetBinding(new
    KeyListModelWigetBinding("personId"));

    section.addSection(generateDescriptionRationaleSection());

    return section;
}

```

### (CM 2.0) 3.3.5 Removing a Section

To remove a section from a view or another section, you need to remove the `section.addSection(Section section)` statement by commenting it out in the institution `CourseConfigurer.java` file. Removing a section removes any subsections for fields within it from the view, so make sure that they are either unnecessary or accounted for elsewhere in the application.

The example below illustrates how to remove the Instructors section from the Course Logistics section.

```

public SectionView generateCourseLogisticsSection() {
    VerticalSectionView section =
        initSectionView(CourseSections.COURSE_LOGISTICS,
LUConstants.LOGISTICS_LABEL_KEY);

    section.setInstructions(getLabel(LUConstants.LOGISTICS_LABEL_KEY + "-instruct") +
"<br><br>");

    section.addSection(generateSchedulingSection());
    //section.addSection(generateDurationSection());
    section.addSection(generateLearningResultsSection());
    section.addSection(generateCourseFormatsSection());
    return section;
}

```

### (CM 2.0) 3.3.6 Adding a Section

You can add sections to parent sections by editing your institution CourseConfigurer file. Adding a section to another section will create a subsection. The section that you are adding may contain fields, other sections, or some combination of fields and sections.

The example below shows how the Course Information view is included on the Course Proposal layout, how the Description and Rationale section is included in the Course Information section, and finally how the Course Description field is included within the Description and Rationale section.

```
public void configure(final CourseProposalController layout) {
    if (modelDefinition.getMetadata().isCanEdit()) {
        addCluStartSection(layout);
        String sections = getLabel(LUConstants.COURSE_SECTIONS);

        layout.addMenu(sections);

        //Course Content
        layout.addMenuItem(sections, generateCourseInfoSection());
        layout.addMenuItem(sections, generateGovernanceSection());
        layout.addMenuItem(sections, generateCourseLogisticsSection());
        .
        .
        .

public SectionView generateCourseInfoSection() {
    VerticalSectionView section = initSectionView(CourseSections.COURSE_INFO,
LUConstants.INFORMATION_LABEL_KEY);
   addField(section, PROPOSAL_TITLE_PATH,
generateMessageInfo(LUConstants.PROPOSAL_TITLE_LABEL_KEY));
    addField(section, COURSE + "/" + COURSE_TITLE,
generateMessageInfo(LUConstants.COURSE_TITLE_LABEL_KEY));
    addField(section, COURSE + "/" + TRANSCRIPT_TITLE,
generateMessageInfo(LUConstants.SHORT_TITLE_LABEL_KEY));
    section.addSection(generateCourseNumberSection());
    FieldDescriptor instructorsFd = addField(section, COURSE + "/" + INSTRUCTORS,
generateMessageInfo(LUConstants.INSTRUCTORS_LABEL_KEY));
    instructorsFd.setWidgetBinding(new KeyListModelWidgetBinding("personId"));
    section.addSection(generateDescriptionRationaleSection());
    return section;
}

.

.

protected VerticalSection generateDescriptionRationaleSection() {
    SectionTitle title = getH4Title(LUConstants.PROPOSAL_TITLE_SECTION_LABEL_KEY);
    VerticalSection description = initSection(title, !WITH_DIVIDER);
    title.setStyleName("cluProposalTitleSection");

    addField(description, COURSE + "/" + "decr" + "/" +
RichTextInfoConstants.PLAIN, generateMessageInfo(LUConstants.DESCRIPTION_LABEL_KEY));
    addField(description, "proposal/rationale",
generateMessageInfo(LUConstants.PROPOSAL_RATIONALE_LABEL_KEY));
    return description;
}
```

## (CM 2.0) 3.4 Configuring Fields within the User Interface

Most field configuration is done in the Service Dictionary as described in Chapter 4. Configuring the Service Dictionaries. Manipulating the

location of the field on the user interface, however, is implemented through the same process as manipulating sections.

As long as the field is properly defined in the data dictionary, you can add a field to a section by specifying it in the institution *CourseConfigurer* file with the appropriate field key. This includes fields which are added through dynamic attributes. Fields are presented in the order in which they appear in the *CourseConfigurer* file and are removed simply by removing the corresponding statement from the section.

For example, the following excerpt from *CourseConfigurer* illustrates the Duration section comprised of the duration type and duration quantity.

```
protected VerticalSection generateDurationSection() {
    VerticalSection duration
    initSection(getH3Title(LUConstants.DURATION_LITERAL_LABEL_KEY),
    WITH_DIVIDER);

    duration.setInstructions(getLabel(LUConstants.DURATION_LITERAL_LABEL_KEY +
"-instruct"));
    GroupSection duration_group = new GroupSection();

    addField(duration_group, COURSE + "/" + CreditCourseConstants.DURATION
+ "/" + "atpDurationTypeKey",
    generateMessageInfo(LUConstants.DURATION_TYPE_LABEL_KEY));

    addField(duration_group, COURSE + "/" + CreditCourseConstants.DURATION
+ "/" + "timeQuantity",
    generateMessageInfo(LUConstants.DURATION_QUANTITY_LABEL_KEY));

    duration.addSection(duration_group);
    return duration;
}
```

## (CM 2.0) 3.5 Configuring Labels, Messages, and Help Text

You can change and add the text for the labels and other messages by updating your messages by updating the KSMG\_MESSAGE table

For example this updates the application title.

```
update KSMG_MESSAGE
set MSG_VALUE = 'My School CM'
where id in (select id from KSMG_MESSAGE where msg_id like 'application%')
```

## (CM 2.0) 4. Configuring the Service Dictionaries

In KS Curriculum Management, the service dictionaries are used to define constraints around service data and to extend the data schema with the use of *dynamic attributes*. Service dictionaries are also used by the KS user interface to generate metadata required for displaying constraints in the user interface and performing client-side validation.

Service data is configured by overriding Spring beans (Java objects and properties) in the corresponding service dictionary.

What	How	Who (Skill Sets)	Level of Difficulty
Service Dictionary	Dictionary XML (Spring bean override pattern)	Business analyst Java developer	Level 2

The Service dictionaries are in lum-impl/src/resources:

- ks-courseinfo-dictionary-context.xml
- ks-lolinfo-dictionary-context.xml
- etc...

## (CM 2.0) 4.1 Dictionary Schema

The dictionary schema is specified using XML Schema Definition (XSD). The root data structure for an object in the Service dictionary is the ObjectStructureDefinition, which is detailed at <https://wiki.kuali.org/display/STUDENTDOC/objectStructureDefinition+Structure>. The structure contains a list of FieldDefinitions (using the attributes tag). Each field definition has the field's name and constraints that applies to it.

### (CM 2.0) 4.1.1 The Spring Bean Schema

A Spring bean is a Java object consisting of an object and its properties. The dictionary is an extension to the Spring bean schema, so you can define beans using Spring and Spring-like syntax. For more information about Spring beans, see <http://static.springsource.org/spring/docs/2.5.x/reference/beans.html>.

Most tags defined in the dictionary-extension.XSD schema have the following Spring attributes:

- ***id***
  - The id is a unique identifier of a bean within a context. The id is used only in the Spring context to refer to other beans and is not related to the path attribute defined for many of the dictionary elements.
- ***abstract***
  - The abstract attribute defines whether this bean is abstract. An abstract bean cannot be instantiated and serves as a template for a bean
- ***parent***
  - The parent attribute references another bean that acts as a ?template?. The parent bean can be either abstract or non-abstract. The parent bean may also have a parent bean, allowing inheritance to be chained through many parent-child relationships.
- ***References***
  - References allow a bean to be reused in multiple places. For most elements, child beans can be defined inline or as references. For example, the field definition for alternatetdentifier in CluInfo attributes is defined through a ref (reference) tag. A separate bean with the id specified in the ref tag defines the constraints around alternatetdentifier. This allows the alternatetdentifier bean to be referenced in other locations, like CourseCrossListings.
- ***Imports***
  - Spring allows a context to be broken up into multiple files using the <import> tag to specify the file to be imported. If there are two or more bean definitions with the same id, the last definition is the definition that is applied. The order of import statements is important. For example, if three separate files are imported and they all have the definition for a bean with an id of ?xyz?, the definition in the last file to be imported will be used.

### (CM 2.0) 4.1.2 Parent-Bean Pattern

ObjectStructureDefinition, FieldDefinition and most Constraint beans are defined at the root level rather than inline, with a reference to a parent abstract bean. The abstract bean contains the template for the object, field or constraint defined. To see how this is implemented in KS Curriculum Management, see the /ks-lum/ks-lum-impl/src/main/resources/ks-cluInfo-dictionary-context.xml file, which defines the basic structural definition for CluInfo objects.

```
<bean id="org.kuali.student.lum.lu.dto.CluFeeInfo-parent"
abstract="true" parent="objectStructureDefinition">
<property name="name" value="org.kuali.student.lum.lu.dto.CluFeeInfo"/>
<property name="attributes">
<list>
<ref bean="cluFeeInfo.cluFeeRecords"/>
<ref bean="cluFeeInfo.descr"/>
<ref bean="cluFeeInfo.id"/>
</list>
</property>
<property name="hasMetaData" value="true"/>
</bean>
<bean id="org.kuali.student.lum.lu.dto.CluFeeInfo"
parent="org.kuali.student.lum.lu.dto.CluFeeInfo-parent"/>
```

The parent-bean pattern allows you to define an overriding bean (with the same Id as KS bean) for your institution. The overriding bean inherits from KS abstract parent bean and, in the process, inherits the template and basic constraints which you can change.

## (CM 2.0) 4.2 Configuring Constraints

Constraints are defined in the FieldDefinition bean as a list of constraint-specific properties.

## (CM 2.0) 4.2.1 Constraint Properties

A field definition can have one or more of the following properties:

- **inclusiveMax**
  - the maximum acceptable value for a numeric field
- **exclusiveMin**
  - the minimum acceptable
- **minOccurs**
  - how many instances of the field must have a valid value for the field to validate
- **maxOccurs**
  - the maximum number of instances of a field allowed
- **minLength**
  - the minimum number of characters that a field must have
- **maxLength**
  - the maximum number of characters that a field may have
- **validChars**
  - identifies specific characters that are allowed to be entered into the field, either by listing individual characters or by using a regular expression prefixed with *regex*:
- **requireConstraint**
  - a list of other fields that must have valid values for the field (with the requireConstraint) to be valid

```
<bean id="addressline2">
<property name="requireConstraint">
<list>
<bean parent="requiredConstraint">
<property name="fieldPath" value="addressline1" />
</bean>
</list>
</property>
</bean>
```

- **caseConstraint**

- a logical constraint that compares the value of a defined field with values in the nested whenConstraint. When the condition is evaluated to true, the properties defined within the whenConstraint will override the defaults. Valid operator values are *equals*, *greater\_than*, *greater\_than\_equal*, *less\_than\_equal*, and *less\_than*.

For example, the following constraint specifies that a value is required for the field if the state is 'ACTIVE' or 'RETIRED' or 'SUBMITTED'

```

<property name="caseConstraint">
<bean parent="caseConstraint">
<property name="operator" value="EQUALS"/>
<property name="fieldPath" value="state"/>
<property name="caseSensitive" value="false"/>
<property name="whenConstraint">
<list>
<bean parent="whenConstraint">
<property name="values">
<list>
<value>SUBMITTED</value>
<value>ACTIVE</value>
<value>RETIRED</value>
</list>
</property>
<property name="constraint">
<bean parent="baseComplexRequired"/>
</property>
</bean>
</list>
</property>
</bean>
</property>

```

- **occursConstraint**

- a dependency constraint that specifies the minimum and maximum number of fields within a set of fields that must be valid. occurs and required attributes may be nested within other occur attributes to define complex dependencies (see the example below).

For example, when entering an address, you may want to require either the zip code or both the city and state. The following illustrates how this would be done with nested occurs and required properties.

```

<!--one of the child occurs or requires must be true and at most they both
must be true-->
<property name="occursConstraint">
<bean parent="occursConstraint">
<property name="min" value="1"/>
<property name="max" value="2"/>
<property name="requiredFields">
<list>
<bean parent="requiredConstraint">
<property name="fieldPath" value="zipcode"/>
</bean>
</list>
</property>
<property name="occursConstraint">
<bean parent="occursConstraint">
<property name="min" value="2"/>
<property name="max" value="2"/>
<property name="requiredFields">
<list>
<bean parent="requiredConstraint">
<property name="fieldPath" value="city"/>
</bean>
<bean parent="requiredConstraint">
<property name="fieldPath" value="state"/>
</bean>
</list>
</property>
</bean>
</property>
</bean>
</property>

```

- **customValidatorClass**

- class name for the custom validator that will be invoked in the process of validating the field. More information on how to implement custom validators can be found in section 4.2.3 Writing Custom Validators.

## (CM 2.0) 4.2.2 Base Template Beans

Base template beans are used by all dictionaries and are maintained in the file `/ks-core/ks-core-impl/src/main/resources/ks-base-dictionary-contains.xml`. Base templates for ObjectStructureDefinitions, FieldDefinitions and Constraints serve as building blocks for other beans through inheritance.

```

<bean id="baseComplex" parent="baseFieldDefinition" abstract="true">
<property name="dataType" value="COMPLEX" />
</bean>
<bean id="cluInfo.descr-parent" abstract="true" parent="baseComplex">
<property name="name" value="descr"/>
<property name="dataObjectStructure" ref="org.kuali.student.core.dto.RichTextInfo"/>
</bean>

```

## (CM 2.0) 4.2.3 Writing Custom Validators

KS Curriculum Management allows institutions to define custom validations and associate them with fields. This provides institution with a mechanism to inject institution-specific logic into KS Curriculum Management without having to touch the source code.

1. Create a custom validator that implements `org.kuali.student.validator` interface and provide implementation for the method:

```

public List<ValidationResultInfo> validateObject(FieldDefinition
field, Object data, ObjectStructureDefinition
objStructure, Stack<String> elementStack);

```

where,

*FieldDefinition field* is the field on which the validation is called

*Object data* is the object on which the validation is called on. This object should contain the value for the field.

*ObjectStructureDefinition objStructure* is the dictionary definition of the Object.

*Stack<String> elementStack* is a stack of field names processed so far. This can be used in generation of path to the field.

2. Set appropriate message key and path in the validation result to ensure that the UI will display the error on the right screen.
3. Override '*lumValidatorFactory-parent*' bean and define a bean with Id '*lumValidatorFactory*' in local context file. Merge the property '*validatorList*' and add the definition to the custom validator bean.

```

<bean id="lumValidatorFactory" parent="lumValidatorFactory-parent" >
<property name="validatorList">
<list merge="true">
<!-- Add Custom Validators Here -->
<bean class="my.inst.customValidator" />
</list>
</property>
</bean>

```

- Finally, add the *customValidatorClass* property to the field that has to be validated. Set the value as the class name of the *customValidator*.

```

<bean id="fieldX" parent="fieldX-parent" >
<property name="customValidatorClass" value="org my.inst.customValidator" />
</bean>

```

#### (CM 2.0) 4.2.4 Overriding KS Configurations

You can override KS bean definitions by specifying a redefinition in a separate file and updating the configuration to reference that file. If you override a bean that is used in multiple places, the definition will be changed for every bean that refers to it.

The following are the steps necessary to override a bean

You can override KS bean definitions by specifying a redefinition in a separate file and updating the configuration to reference that file. If you override a bean that is used in multiple places, the definition will be changed for every bean that refers to it.

The following are the steps necessary to override a bean

1. Create a new file that contains your override (e.g., *MyOverride-context.xml*).
2. Redefine the bean with your new rules. Make sure the *id* attribute matches the *id* of the bean you want to override. For example, the following overrides a constraint named *lineText* to allow the equal sign (=) and not allow tabs:

```

<bean id="lineText" parent="ValidCharDefinition">
<property name="value"
value="regex:[A-Za-z0-9.,&gt;&lt;_\\/\\-\\?\\+=()\\[\\];:\\\";\\$#@!= ]*"/>
</bean>

```

3. Update *ks-lum-config.xml* and edit the *ks.lum.dictionary.serviceContextLocations* property in your configuration to reference the new context file, ensuring that the new file appears last so that the override beans will take precedence over the beans defined before it.

```

<param
name="ks.lum.dictionary.serviceContextLocations">classpath:ks-courseInfo-dictiona
ry-context.xml;classpath:ks-programInfo-dictionary-context.xml;classpath:ks-state
ment-dictionary-context.xml;classpath:MyOverride-context.xml</param>

```

## (CM 2.0) 4.3 Adding a Field (Dynamic Attributes)

Almost every message data structure has dynamic attributes called attributes, which allows you to add data not specified directly in the Service Contract. The dynamic attributes are a set of key/value pairs that you can use to make extensions to the data.

While they are represented as if they are direct fields within the object described, a consumer is expected to recognize the standard approach to dynamic attributes and wrap them accordingly. There will also be corresponding changes required in the application and user interface, such as label text, hidden fields, etc.

Follow these steps to add a dynamic field to the service dictionary. The examples included in the procedure illustrate how to add *finalExamStatus* and *finalExamRationale* fields to the *CourseInfo* object structure.

1. Create a custom dictionary extension file (e.g., *myinst-courseInfo-dictionary-context.xml*) and import the KS dictionary file that contains the definition of the object structure to which the new field must be added (*import ks-courseInfo-dictionary-context.xml*).  
If a field is being added to a business service object structure with a corresponding entity service object, then no change is required at the entity service level. The system will automatically map dynamic attributes from the business service to its entity service.
2. Define your override bean with the same Id as the object structure and set the parent bean as the parent of the corresponding KS bean.

```

<bean id="org.kuali.student.lum.course.dto.CourseInfo"
parent="org.kuali.student.lum.course.dto.CourseInfo-parent">
<property name="attributes">
<list merge="true">
<ref bean="customCourseInfo.finalExamStatus" />
<ref bean="customCourseInfo.finalExamRationale" />
</list>
</property>
</bean>

```

- Note:** The list tag within the *attributes* property has *merge* set to "true." This adds the two fields to the list of fields defined in the parent bean.
3. Create bean definitions for the newly added fields with the constraints that apply to them. The *dynamic* property for these bean definitions should be set to "true."

```

<bean id="courseInfo.finalExamStatus" parent="baseString">
<property name="name" value="finalExamStatus" />
<property name="minOccurs" value="0" />
<property name="maxOccurs" value="1" />
<property name="maxLength" value="15" />
<property name="validChars" ref="alpha"/>
<!-- Maybe a lookup constraint to restrict the values -->
<property name="dynamic" value="true"/>
</bean>
<bean id="courseInfo.finalExamRationale" parent="baseString">
<property name="name" value="finalExamRationale" />
<property name="dynamic" value="true"/>
<property name="minOccurs" value="0" />
<property name="maxOccurs" value="1" />
<property name="maxLength" value="250" />
<property name="validChars" ref="multiLineText"/>
</bean>

```

4. Update the appropriate `serviceContextLocations` property in your config file and add the name of the override dictionary file at the end.

```

<param name="ks.lum.dictionary.serviceContextLocations">
classpath:ks-courseInfo-dictionary-context.xml;classpath:ks-programInfo-dictionar
y-context.xml;
classpath:ks-statement-dictionary-context.xml;classpath:myinst-courseInfo-diction
ary-context.xml
</param>

```

No changes are required in the service implementation to add the fields. The `MetadataserviceImpl` and `DataMapper` used by KS User Interface Framework will automatically obtain the new fields and present them as regular fields for the object to the application layer. These fields could then be added to the UI using the steps defined in Section 3.4 *Configuring Fields within the User Interface*.

## (CM 2.0) 5. Configuring Searches, Pickers, and Browses

KS Curriculum Management supports three primary mechanisms for finding data:

- Standard searches, such as Find Course and Find Major are provided on the main menu.
- A picker takes search criteria and displays search results from which the user can select the desired value. The select value is usually used to populate the value of another field.
- A browser, such as the Catalog Browser, allows the user to select the result of a search as the criteria for another search.

There are several aspects of searches that you may want to change:

- What criteria needs to be supplied for a particular Search
- What information is returned in the results
- How the results are ordered
- How the search is performed
- Who is allowed to run a search
- What Standard Searches are available where

All three mechanisms use the same infrastructure, which is composed of four main parts as listed below. Configuring a search involves coordinating changes for all of these parts.

- Search Types - Formal definitions of searches in the KS Curriculum Management
- Fields - Regular database and freestanding fields defined in the dictionary
- Lookup Definitions – A lookup definition configures a search for a particular situation on the user interface, defining how criteria and results are used.

- Lookup Field Bindings – A field binding specifies the field within the user interface that will use the lookup definition.

## (CM 2.0) 5.1 Before You Begin

Configuring a search will require changes to a variety of dictionary XML files. To avoid changes being overwritten in subsequent releases of KS Curriculum Management, you will need to create copies of these files in your institution configuration project.

Within your institution configuration project (see *Chapter 2. Setting Up an Institution Configuration Project*), create an institution XML file (e.g., src/main/resources/cdm-lum-ui-lookup-context.xml) that imports the Dictionary XML at ks-core-impl/src/main/resources/lum-ui-lookup-context.xml.

### Search Types

Because Kuali Student is service-based it does not allow direct access to the underlying database. Instead Kuali Student allows access through formally defined searches which describe how to execute the calls to the underlying database to get the requested data. These definitions are known as a SearchType which is a named definition comprised of the following:

- SearchCriteriaType – defines what parameters may be supplied for searching.
- SearchResultType – defines which columns are returned in the results.
- The SearchType definition also defines the logic used to execute the search.

The predefined searches that are provided with KS Curriculum Management and are defined in the following locations:

- **ks-core-impl /src/main/resources**
  - em-search-config.xml - enumeration management "codes" service searches
  - proposal-search-config.xml – Proposal Service searches
  - atm-search-config.xml – Academic Time Period Service searches
  - comment-search-config.xml – Comment and tagging service searches
  - organization-search-config.xml – Organization Service searches
- **ks-lum-impl/src/main/resources**
  - lu-search-config.xml – Learning Unit Service searches (courses and programs)
  - lo-search-config.xml – Learning Objectives Service searches
  - lrc-search-config.xml – Learning Results Catalog Service searches
- **ks-core-impl/src/main/org/kuali/student/core/personsearch/service/impl**
  - QuickViewByGivenName.java – A hardwired search that calls the KIM person search mechanism

The search configurations are maintained in XML, but you see formatted versions of them here. These formatted versions are artifacts of unit testing and are provided for reference. The unit tests that created these files are:

- ks-core-impl
  - TestSearchConfig.java
- ks-lum-impl
  - TestSearchConfigs.java

## (CM 2.0) 5.2 Configuring Search Criteria and Results

A search configuration is a named set of data combined with a JPQL statement. Together, these elements define the method to use specific criteria to return a set of results. Search configurations (and the criteria and results components) can be reused throughout the system. The predefined searches in KS Curriculum Management should suit most needs. You can modify an existing search to use additional search criteria or return an additional results column.

To add or remove the criteria and/or results for the search, edit the corresponding cdm-xxx-search-config file in your institution configuration project.

The following is an example of the JPQL statement for a search configuration named org.search.generic:

```
SELECT org.id, org.shortName, org.longName, org.type.id, From Org org
```

The search accepts a number of optional criteria and returns the ID, short name, long name and type for each organization that matches the criteria. The WHERE clause is not included, allowing for parameters to be optional. The WHERE clause is built depending on the parameters that are passed. Parameter matching is assumed to be text fuzzy searches unless overridden.

The search definition specifies what parameters are allowed and what results columns are returned by the search. How the search is displayed, and what parameters are presented in the user interface, is configured in the lookup for the specific use of the search.

### (CM 2.0) 5.2.1 Example of Adding a New Criteria Parameters to the Search Type

Query parameters must be defined in the corresponding search before you can configure them. For example, to add a "Division" field to the search, you would make the following changes to the lu-search-config.xml file:

1. Define the parameter:

```

<search:queryParams>
<search:queryParam id="lu.queryParam.luOptionalDivision"
    parent="lu.queryParam.luOptionalDivision-parent"/>
<search:queryParam id="lu.queryParam.luOptionalDivision-parent"
    optional="true" abstract="true">
<ref bean="field.cluInfo.officialIdentifier.division.fd"/>
</search:queryParam>

```

2. Insert the parameter into the search criteria (added text in bold):

```

<search:searchCriteriaTypeInfo id="lu.criteria.luAdvancedCriteria-parent"
abstract="true">
<search:queryParams>
<ref bean="lu.queryParam.luOptionalCode"/>
<ref bean="lu.queryParam.luOptionalDivision"/>
<ref bean="lu.queryParam.luOptionalLevel"/>

```

3. Map the division parameter to JPQL:

```

<entry key="lu.queryParam.luOptionalDivision">
    <value>clu.officialIdentifier.division</value>
</entry>

```

Similarly, you must define the result columns in the corresponding search by doing the following:

1. Formally define the result column.
2. Add the result column to the search.
3. Add the result column to the JPQL *select* clause.

### (CM 2.0) 5.2.2 Cross Service Searches

Cross service searches are used to join information that exists in two searches. Since KS Curriculum Management is service-oriented, this is done by crossing database joins. The data from one search is used to select and return data from another search similar to an SQL sub-query.

### (CM 2.0) 5.3 Field Definitions

All searches, lookups and browses must be connected to a field so that the user interface can invoke the search. The field must be defined in the dictionary before a lookup can be attached to it.

There are three different types of fields that may have lookups attached to them:

- Regular fields that already exist because they are defined as part of the service contract.
- A field defined using dynamic attributes (see [4.3 Adding a Field \(Dynamic Attributes\)](#)).
- Freestanding search/browse fields that you want to use to find data.

Dynamic attributes can be defined in the following files:

- **ks-lum-impl/src/main/resources**
  - ks-courseInfo-dictionary-context.xml – Course fields
  - ks-programInfo-dictionary-context.xml – Program fields
  - ks-cluset-ui-object-dictionary-context.xml – CLU set fields
- **ks-lum-impl/src/main/resources**
  - ks-proposalInfo-dictionary-context.xml --Proposal fields
  - ks-statement-dictionary-context.xml – ules statements, pre-requisites and requirements
  - ks-workflow-dictionary-context.xml -- collaborators

Free standing search/browse fields for the curriculum management application are defined in  
 ks-lum-impl/src/main/resources/ks-lu-search-dictionary-context.xml

**IMPORTANT:** The lookups defined in these dictionary files are used only for validation and does not affect the user interface. In KS Curriculum Management, searches that define what is valid are separate from searches that define how the user searches for those field fields in a particular context in the user interface. If you want to reuse the validation lookups in the user interface, you can define them for validation in a common file: ks-common-impl/src/main/resources/common-lookup-context.xml. This file is imported at the top of the user interface lookup XML file, so these definitions can be reused as the basis of user interface configurations of lookups.

Formatted examples of these lookups are available at [\(CM 2.0\) Formatted View of CLU Searches](#). These formatted versions are artifacts of unit testing and are provided for reference only. The unit tests that created these files are:

- **ks-core-impl**
  - TestProposalDictionary.java
  - TestStatementDictionary.java
  - TestWorkflowDictionary.java
- **ks-lum-impl**
  - TestProgramDictionary.java
  - TestCluInfoDictionary.java
  - TestClusetUiObjectDictionary.java
  - TestCourseInfoDictionary.java
  - TestFreestandnigLuSearchDictionary.java

## (CM 2.0) 5.4 Lookup Definitions and Lookup Field Bindings

Lookup definitions configure a search for a particular instance in the user interface. A lookup definition can change how criteria and results are used in a particular instance---it cannot add new criteria or results to a search.

The lookup must be defined before binding it to a field. Both the lookup definitions and the field bindings are defined in the same XML file:

- ks-lum-ui/src/main/resources
  - lum-ui-lookup-context.xml

You can make the following configuration changes in a lookup definition:

- Criteria Changes
  - provide default value(s) for a search criteria
  - provide a hardcoded value that cannot be changed to a search
  - suppress access to particular search criteria
  - provide a better descriptive label or name for the search criteria
  - Indicate what type of widget should be used to collect the user-entered parameter value
- Result Changes
  - The title of the window
  - The column titles
  - Suppress certain columns in the display of the result
  - Change the order of the columns in the result table
  - Change the sort order of the rows in the result table
  - Change which result column is stored in the data field when the user picks a row
  - Limit the maximum number of rows returned by the search

### ***Configuring a Lookup***

Figure 3 below shows the lookup for an organization search that specifies the label, description, the widget that the user will use to enter the criteria and other attributes of lookup. The result is a suggestion box widget that suggests organization entities by matching the organization's long name to the text that the user enters, and returns the orgID for the item the user selects from the results.

```

<bean
id="kuali.lu.lookup.admin.departments.suggest" class="org.kuali.student.core.assembly.
data.UILookupData" parent="kuali.common.lookup.admin.departments.suggest">
<property name="id" value="kuali.lu.lookup.admin.departments.suggest" />
<property name="name" value="Organization search" />
<property name="desc"
          value="Search for administrative department organization" />
<property name="widget" value="SUGGEST_BOX" />
<property name="searchParamIdKey" value="org.queryParam.orgOptionalId" />
<property name="resultDisplayKey" value="org.resultColumn.orgOptionalLongName" />
<property name="resultSortKey" value="" />
<property name="results">
<list>
<bean parent="result">
<property name="key" value="org.resultColumn.orgId" />
<property name="name" value="Organization Identifier" />
<property name="desc" value="Identifier for the organization" />
<property name="dataType" value="STRING" />
<property name="hidden" value="true" />
</bean>
<bean parent="result">
<property name="key" value="org.resultColumn.orgOptionalLongName" />
<property name="name" value="Name" />
<property name="desc"
          value="Long name for the organization, recorded as the default
listing." />
<property name="dataType" value="STRING" />
<property name="hidden" value="false" />
</bean>
</list>
</property>
</bean>
```

**Figure 3: Lookup for an Organization Search**

### (CM 2.0) 5.4.1 Constraints

org.kuali.student.core.assembly.data.ConstraintMetadata contains the constraint object structure.

Items with an asterisk (\*) are not implemented at this time.

Attribute	Type	Description
<ConstraintMetadata> childConstraints;	LIST	A list of child constraints (used to group constraints).
comments –	STRING	Developer comments for this constraint.
desc	STRING	A text description of the constraint.
Id	STRING	A unique identifier for the constraint.
maxLength;	INTEGER	The maximum character length for the field.
maxOccurs;	INTEGER	The maximum number of times the field can occur (if > 1, it is assumed this is a list of items).
maxValue;	STRING	The maximum value that a numeric value can be.

messageId –	STRING	The message ID to use for message translation.
minLength;	INTEGER	The minimum character length for the field.
minOccurs;	INTEGER	The minimum number of times the field can occur. Usually 0 (indicates the field is optional) or 1 (indicates the field is required), but can be higher if multiple values are required.
minValue;	STRING	The minimum value that a numeric value can be.
*serverSide –	BOOLEAN	Indicates whether validation should be done on the server (otherwise, only on client).
*specialValidator –	STRING	Identifies a special class to use for validation.
validChars	STRING	Identifies specific characters that are allowed to be entered into the field, either by listing individual characters or by using a regular expression prefixed with <i>regex</i> :

### (CM 2.0) 5.4.2 Lookup Parameters

org.kuali.student.core.assembly.data.LookupParamMetadata defines the parameters for a search, with default values or searches of their own.

Items with an asterisk (\*) are not implemented at this time.

Attribute	Type	Description
key	STRING	Corresponds to the query parameter key for the search
childLookup	LookupMetadata	Defines whether lookup parameter uses another lookup (for example, a parameter for searching for organizations is a list of organization types). This is also used by the catalog browser to chain together searches in a cascading style.
writeAccess	Metadata	The default write access. One of: ON_CREATE, ALWAYS, NEVER, WHEN_NULL, REQUIRED
dataType	Data.DataType	Defines the datatype of the lookup parameter. One of: STRING, INTEGER, LONG, FLOAT, DOUBLE, BOOLEAN, DATE, TRUNCATED_DATE, DATA, LIST
defaultValueList	STRING ARRAY LIST	A list of values to which this parameter should be set. Only one of defaultValueList or defaultValueString should be specified, but not both. See note for defaultValueString below.
defaultValueString	STRING	A single value to which this parameter should be set. Only one of defaultValueList or defaultValueString should be specified, but not both. This field along with defaultValueList, is used to customize a “generic” search to produce a result set that matches the needs for field field. For example: this field might be set to “org.type.Department” to restrict the search to just departments and not committees, boards, etc.
name	STRING	A text name identifying the parameter.
desc	STRING	A text description of the parameter.

widget	VALUE	Identifies the widget to use to display the parameter in the view. One of: NO_WIDGET, SUGGEST_BOX, ADVANCED_LIGHTBOX, DROP_DOWN, BUTTON, CHECKBOX_LIST
--------	-------	--------------------------------------------------------------------------------------------------------------------------------------------------------

### (CM 2.0) 5.4.3 Lookup Results

org.kuali.student.core.assembly.data.LookupResultMetadata defines the results for a search, mapping the search results to the fields displayed in the view.

Attribute	Type	Description
key	STRING	Corresponds to the searchResultKey from the search. This must match the resultKey in the corresponding search.
name	STRING	A text name identifying the search results. This name is what appears in the column header in the table of search results.
desc	STRING	A text description of the search results.
dataType	DATA.DataType	The type of data returned. One of: STRING, INTEGER, LONG, FLOAT, DOUBLE, BOOLEAN, DATE, TRUNCATED_DATE, DATA, LIST
Hidden	BOOLEAN	Whether the result be shown in the list of results. Often the id field is returned but not displayed to the user.

### (CM 2.0) 5.5 Field Binding

When binding a lookup to a field, the field path must exactly match the object.field definition in the main dictionary. Figure 4 binds that configuration to the unitsDeployment field on the courseInfo object.

```

<bean id="course.unitsDeployment" parent="department-lookup-binding-parent">
    <property name="path" value="courseInfo.unitsDeployment" />
</bean>

<bean id="department-lookup-binding-parent" parent="uilookupconfig-parent"
    abstract="true">
    <property name="name" value="department.lookup" />
    <property name="type" value="kuali.reqComponent.field.type.person.id" />
    <property name="initialLookup" ref="kuali.lu.lookup.admin.departments.suggest" />
    <property name="additionalLookups">
        <list>
            <ref bean="kuali.lu.lookup.admin.departments.advanced" />
        </list>
    </property>
</bean>
```

Figure 4: Lookup Field Binding for an Organization Search

### (CM 2.0) 5.5.1 Type Aware Field Binding

It is also possible to configure different lookups depending on the type of the object. To do this you simply add a type property like this

```
<property name="path" value="reqCompFieldInfo.value" />
<property name="type" value="kuali.reqComponent.field.type.person.id" />
```

When the UI gets the metadata it must supply this type as a parameter and then this lookup is bound to that field.

### (CM 2.0) 5.5.2 Configuring a Search-Select Widget for a Lookup

The search/select widget uses the lookup metadata to configure the search behavior, including which results fields to display, filters applied to the search box and suggest box, etc. Note that null for FieldDescriptor in the example below causes the label to default to what is defined in the Service Dictionary.

```
QueryPath path = QueryPath.concat("course", "department");
Metadata meta = modelDefinition.getMetadata(path);
FieldDescriptor fd = new FieldDescriptor(path.toString(), null, meta);
Section.addField(fd)
```

*Figure 5: A Search>Select Widget Configuration*

## (CM 2.0) 5.6 Configuring Enumerations

Enumerations are lists of possible codes and values (for example, the list of campus locations, subject areas, states, provinces and countries, etc.) from which the user selects an appropriate value for a field. The set of values is maintained independently in the database and is not derived directly from information in another service.

### (CM 2.0) 5.6.1 Adding or Changing Enumeration Value Sets

Enumerations are maintained directly in the database and can be added or changed using SQL commands.

### (CM 2.0) 5.6.2 Configuring a Widget to Use an Enumeration

You can attach an enumeration to a widget and specify parameters and results the same way you configure Search>Select. The enumeration management service has a search type defined that returns the list of codes and values as a standard search result, which can be configured the same as any other search. See [5.5.2. Configuring a Search>Select Widget for a Lookup](#).

## (CM 2.0) 5.7 Configuring the Catalog Browser

The catalog browser uses a cascading search model. It uses the results of one search to select the data for the next search---to allow a user to navigate and display courses that are in the catalog. The searches can be cascaded to a depth of many levels.

Two cascading searches are provided with KS Curriculum Management:

- Search by Subject Area
  - The search initially returns all subject areas.
  - The user selects a subject area to return courses in that subject area.
- Search by School or College
  - The search initially returns a list of all schools or colleges.
  - The user selects a school or college to return a list of departments.
  - The user selects a department to return a list of courses for that department

### (CM 2.0) 5.7.1 Displaying the Catalog Browser in the User Interface

Since the catalog browser uses the same underlying configuration mechanism as pickers it has to have a “field” to which configuration is attached. However, for the catalog browser the field is not attached to a widget so it isn’t actually displayed on the screen. Instead, the field is used only to manage the configuration for the catalog browser. The field is neither stored nor updated.

In the KS Curriculum Management, the fields that have been defined are:

BrowseCourseCatalog

- bySubjectArea
  - courseid
- bySchoolOrCollege
  - courseid

These fields are defined in the Service Dictionary. The difference between the definitions are the corresponding lookups, detailed in the following sections.

### (CM 2.0) 5.7.1.1 Searches and Lookups for Browse by Subject Area

Browse by Subject Area has two defined searches (defined queries) with corresponding Lookups (configured searches): Searches are defined in the corresponding XXXX-search-config.xml files and the lookups are defined along with the field in the lum-orchestration-dictionary.xml

Search	Lookup
<b>enumeration.management.search</b> - a generic query that allows you to get a list of enumerated values, for example States or Countries or in this case a list of valid subject areas.	<b>Kuali.lu.browse.AllSubjectAreas</b> Configured by setting defaultValue to: <ul style="list-style-type: none"><li>Enumeration Type =Subject Areas.</li></ul>
<b>lu.search.generic</b> - a generic query that allows you to search for CLUs	<b>CourseCatalogBySubjectArea</b> Configured by setting defaultValue to: <ul style="list-style-type: none"><li>Type = Credit Course</li><li>State = Activated</li><li>Subject Area = The subject area selected by the user in the previous search</li></ul>

### (CM 2.0) 5.7.1.2 Searches and Lookups for Browse by School or College

Browse by School or College has three defined searches (defined queries) with corresponding Lookups (configured searches):

Search	Lookup
<b>Org.search.generic</b> – a generic search for organizations.	<b>Kuali.lu.browse.schoolsAndColleges</b> Configured by setting defaultValue to: <ul style="list-style-type: none"><li>Org Type = either a School or College</li></ul>
<b>org.search.orgQuickViewByRelationTypeRelatedOrgTypeOrgId</b> – a generic query that allows you to search for CLUs.	<b>Kuali.browse.departments.in.school</b> Configured by setting defaultValue to: <ul style="list-style-type: none"><li>Org Type = Department</li><li>Relation Type = Contains</li><li>Parent Org ID = the organization selected by the user in the previous search</li></ul>
<b>lu.search.by.admin.org</b> - a seach query that allows you to search for CLUs by the administrative org and they type of relationship that org has to the course differently.	<b>CourseCatalogBySubjectArea</b> Configured by setting defaultValue to: <ul style="list-style-type: none"><li>Type = Credit Course</li><li>State = Activated</li><li>Primary Admin Org = The department selected by the user in the previous search</li></ul>

### (CM 2.0) 5.7.2 Configuring What is Displayed on the Screen

You can configure how the result table is presented in the user interface. The following sections show examples of the changes you can specify.

Formatted examples are available at [REFERENCE Create public wiki page for [at this location](#) and link to it.]. These formatted versions are artifacts of unit testing and are provided for reference only. The unit tests that created these files are:

- ks-lum-ui
- TestMetadataServiceDictionary.java

#### (CM 2.0) 5.7.2.1 Hide or Show a Column

In this example, the internal Clu ID is hidden in the results:

```

<bean parent="result">
    <property name="dataType" value="STRING" />
    <property name="desc" value="Identifier of a Clu" />
    <property name="hidden" value="true" />
    <property name="key" value="lu.resultColumn.cluId" />
    <property name="name" value="Clu Id" />
</bean>

```

### (CM 2.0) 5.7.2.2 Change the Column Heading

In this example, the column for the long name for a course is changed to “Name”:

```

<bean parent="result">
    <property name="dataType" value="STRING" />
    <property name="desc"
        value="Long name for the lu, recorded as the default listing." />
    <property name="hidden" value="false" />
    <property name="key" value="lu.resultColumn.luOptionalLongName" />
    <property name="name" value="Name" />
</bean>

```

### (CM 2.0) 5.7.2.3 Change the Column Order

To change the order of the columns in the search results, simply rearrange the order in which the result columns are defined in the service orchestration dictionary.

## (CM 2.0) 5.8 Configuring the Program Browser

Unlike the catalog browsers which uses a cascading search model, the program browser is configured to display all available programs in the system and allows the user to further narrow the list using a set of filters (or facets). The set of filters available and the the search used to display all available programs uses the same configuration mechanism as pickers. Controlling the rows and columns displayed for the program search results in the program browser is configured in the same manner as described in earlier parts of section 5.

In KS Curriculum Management, the following search and lookup is defined to control the searches for browse program:

Search	Lookup
<b>lu.search.browseProgram -</b> Query that retrieves all graduate and under graduate and undergraduate academic prorams.	<b>kuali.lu/browseProgram/search</b> Used to bind the search to the browse program widget.

In addition additional searches and lookups are defined which control the facets available in the filter widget for browse academic programs.

### (CM 2.0) 5.8.1 Configuring Facets for Browse Filter

The browse program results table may be narrowed by a set of filters (or facets). A user has the ability to limit the set of results displayed by selecting a selection criteria from a set of predefined search facets. The set of facets and the selection criteria available for each facet are all controlled by the same mechanism as the pickers. In most cases the facets are simply backed by an enumeration, but they need not be limited to enumerations, they can be any search available in the system. However it should be noted that if the search result backing the facet is large, it can become unwieldy as it will generate a large number of check boxes. It should also be noted that the facet's selection criteria be included in the result set of the data set being browsed, otherwise selecting criterion from the facet will have no effect.

The set of filters are defined using a lookup the lum-ui-lookup-context.xml file which is bound to the filter widget in the screen. Each facet is defined using a lookup definition with the additionalLookups property. Note an initialLookup is defined, but is not actually used. In the following example, the filter has five search facets. To add a new search facet simply add an new lookup definition to the additionalLookups. The order in which they are listed here will be the order in which these facets are rendered on the screen.

Example definition for list of facets available for filter widget

```
<bean id="browseProgram.filter" parent="uilookupconfig-parent">
    <property name="path" value="browseProgram.filter" />
    <property name="initialLookup" ref="kuali.lu/browseProgram/filter/level" />
    <property name="additionalLookups">
        <list>
            <ref bean="kuali.lu/browseProgram/filter/level"/>
            <ref bean="kuali.lu/browseProgram/filter/degreeType"/>
            <ref bean="kuali.lu/browseProgram/filter/credential"/>
            <ref bean="kuali.lu/browseProgram/filter/campus"/>
            <ref bean="kuali.lu/browseProgram/filter/status"/>
        </list>
    </property>
</bean>
```

Example enumeration backed search to define selection criterion for level facet.

```

<bean id="kuali.lu.filter.enumeration-parent" abstract="true">
    <property name="id" value="course" />
    <property name="title" value="Courses"/>
    <property name="widget" value="CHECKBOX_LIST" />
    <property name="resultDisplayKey" value="enumeration.resultColumn.value" />
    <property name="resultSortKey" value="enumeration.resultColumn.sortKey" />
    <property name="resultReturnKey" value="enumeration.resultColumn.code" />
    <property name="usage" value="DEFAULT" />
    <property name="searchTypeId" value="enumeration.management.search" />
</bean>

<bean id="filterParam-parent" parent="param" abstract="true">
    <property name="key" value="enumeration.queryParam.enumerationType" />
    <property name="writeAccess" value="NEVER" />
    <property name="dataType" value="STRING" />
    <property name="optional" value="true" />
    <property name="name" value="Enumeration Type" />
    <property name="desc" value="The type of the enumeration to search" />
    <property name="caseSensitive" value="true" />
</bean>

<bean id="kuali.lu.browseProgram.filter.level"
parent="kuali.lu.filter.enumeration-parent">
    <property name="id" value="lu.resultColumn.cluOfficialIdentifier.level" />
    <property name="title" value="Level"/>
    <property name="resultReturnKey" value="enumeration.resultColumn.code"/>
    <property name="params">
        <list>
            <bean parent="filterParam-parent">
<property name="defaultValueString" value="kuali.enum.lu.program.level" />
            </bean>
            <bean parent="filterParam-parent">
<property name="key" value="enumeration.queryParam.enumerationCode" />
                <property name="defaultValueList">
                    <list>
                        <value>kuali.lu.program.level.UnderGraduate</value>
                        <value>kuali.lu.program.level.Graduate</value>
                    </list>
                </property>
            </bean>
        </list>
    </property>
</bean>

```

Example non-enumeration backed search to define selection criterion for degree type facet.

```

<bean id="kuali.lu.browseProgram.filter.degreeType"
class="org.kuali.student.common.assembly.data.UILookupData">
    <property name="id" value="lu.resultColumn.resultComponentId" />
    <property name="title" value="Degree Type"/>
    <property name="widget" value="CHECKBOX_LIST" />
    <property name="resultDisplayKey" value="lrc.resultColumn.resultComponent.name" />
    <property name="resultSortKey" value="lrc.resultColumn.resultComponent.name" />
    <property name="resultReturnKey" value="lrc.resultColumn.resultComponent.name" />
    <property name="usage" value="DEFAULT" />
    <property name="searchTypeId" value="lrc.search.resultComponent" />
    <property name="params">
        <bean parent="param">
            <property name="key" value="lrc.queryParam.resultComponent.type" />
            <property name="writeAccess" value="NEVER" />
            <property name="dataType" value="STRING" />
            <property name="optional" value="true" />
            <property name="name" value="Degree Type" />
            <property name="desc" value="Degree Type" />
            <property name="caseSensitive" value="true" />
            <property name="defaultValueString" value="kuali.resultComponentType.degree" />
        </bean>
    </property>
</bean>

```

## (CM 2.0) 6. Configuring Business Processes and Workflow

Business processes and workflow are administered through KIM (Kuali Identity Management) and KEW (Kuali Enterprise Workflow). This document explains how these are used in KS Curriculum Management---details for how to administer workflow and identities are provided in the Kuali Rice documentation.

What	How	Who (Skill Sets)	Difficulty
Changing a route pattern	KEW (Kuali Enterprise Workflow) document type xml and KIM (Kuali Identity Management) data	Business analyst Java developer	Level 2
Adding users to an existing routing pattern	Roles defined in KIM (Kuali Identity Management)	Business analyst Subject matter expert	Level 2
Authorization	Create permissions and roles in KIM (Kuali Identity Management)	BA	Level 2

Workflow is a business process in which interaction is required in one or more steps. KS Curriculum Management uses Kuali Enterprise Workflow (KEW) to manage approval based workflows (how items are routed based on the document type) and Kuali Identity Management (KIM) to manage who is responsible for processing the item at a node within the workflow.

### (CM 2.0) 6.1 What is NOT Covered in this Document

You can find details of the concepts and procedures of workflow and identity management in the RICE documentation at <http://site.kuali.org/rice/2.0/reference/html/portal.html>. On that site you can find both User Guides and Technical Reference guides for the various modules of Kuali RICE. The KS Curriculum Management Configuration Guide gives you the information you need to understand how KEW and KIM are used in KS Curriculum Management, but not the details of defining complex workflow, adding identities, etc.

**IMPORTANT:** The standard configuration for KS Curriculum Management uses Kuali Rice 2.2.0. When consulting documentation, please make sure you are using Rice 2.2.0.

### (CM 2.0) 6.2 Kuali Workflow Documents and KS Curriculum Management Proposals

In KS Curriculum Management, the processes for some objects, such as Courses, have *proposals* attached to them. A KS *proposal* object is the link to the Kuali Workflow *document* object. The KS proposal signifies that the function being performed (such as a *creation* or a *modification*) on a KS object requires some form of approval through some interaction with the workflow document.

A "Credit Course Modification" is an example of a process in KS Curriculum Management. When a change is made to a credit course, a proposal is linked to a document inside Kuali Enterprise Workflow (KEW). When the proposal is created, the workflow document is created. When the initiator of the change submits the proposal, the workflow document is put into ENROUTE status and the document is routed to the first approval request that is generated.

## (CM 2.0) 6.2.1 Configuring Workflow Document Content and Document Title

Each workflow document has an associated document title and some XML document content, which is used by workflow processing to make routing decisions. The title and content are also used by other parts of workflow, such as Document Search, which uses the Document Search Attributes.

These elements can be configured by modifying the *ProposalWorkflowFilter* Spring bean definition associated with the workflow process. The following is the configuration used for the course proposal process:

```
<bean id="courseProposalWorkflowFilter" parent="parentProposalWorkflowFilter">
    <property name="proposalReferenceType" value="kuali.proposal.referenceType.clu"/>
    <property name="defaultDocType" value="kuali.proposal.type.course.create"/>
    <property name="docTypeConfigs">
        <list>
            <bean
                class="org.kuali.student.core.assembly.transform.DocumentTypeConfiguration">
                <property name="documentType" value="kuali.proposal.type.course.create"/>
                <property name="defaultDocumentTitle" value="Course proposal for
{courseTitle}"/>
                <property name="docContentFieldMap">
                    <map>
                        <entry key="cluid" value="id"/>
                        <entry key="orgId" value="unitsContentOwner/0"/>
                    </map>
                </property>
            </bean>
            ....
        </list>
    </property>
</bean>
```

The *proposalReferenceType* and *defaultDocType* are used by the application to determine which type of proposal to create and what the default workflow document type to associate with the proposal being created, respectively. See section 6.3 Document Types on page 49 for more information about Workflow Document Types.

For each type of workflow Document Type that can be associated with the proposal, a configuration must be defined to provide the document title and the elements to be included in the document content. The *DocumentTypeConfiguration* bean is used to provide these elements and has the following properties:

Property Name	Function
documentType	The name of the workflow Document Type.
defaultDocumentTitle	The value to be used for the document title. This supports property path replacement, using the '{ }' notation. In the example config the '{courseTitle}' would be replaced with the value of the courseTitle field of the course data being persisted.
docContentFieldMap	This is a map of key value pairs which is used to determine which data field properties are to be included in the document content. The key is the name to use for the content's xml element and the value is the property path for the xml elements value. In the example the <entry key="cluid" value="id"/>, will result in the following to be included in the xml document content: <cluid>1234567890123456890</cluid>.

## (CM 2.0) 6.2.2 Configuring Fields Required by Workflow Node

A proposal can be configured to have certain fields required on submission or approval as it moves through nodes in the workflow process. This can be accomplished by making use of the case constraint property available in the service dictionary. (See section 4.2.1 Constraint Properties for more on case constraints). As a proposal object (e.g. Course or Program) moves through the workflow process it remains in DRAFT state. When the proposal object is in DRAFT state the dictionary can be configured to enforce a required constraint based on workflow node. Simply create a top level case constraint that checks to see if the proposal object is in DRAFT state, then use a nested case constraint which applies a required constraint based on the "proposal/workflowNode". In the example below the following case constraint property can be set on any property where we would like the field to be required on submit and then required to save on subsequent workflow nodes. It is important to note that the first value defined in the list of workflow node values is important in determining if the field is simply required to save for that node or if it is required to submit or approve. In the example below since PreRoute is the first node defined, the field will be required to submit and required to save for all subsequent nodes. If on the other hand "Document Organization Review" was the first node defined it would be required to approve for that node and required to save for all subsequent nodes.

NOTE: A limitation of Kuali Student 1.2 is that one can only define one set of workflow based constraints per object. This means if more than one workflow document type exists for a proposal object (e.g. create course and modify course) then the same workflow node constraints will be applied to both workflow documents.

```
<property name="caseConstraint">
    <bean parent="caseConstraint">
        <property name="operator" value="EQUALS"/>
        <property name="fieldPath" value="state"/>
        <property name="caseSensitive" value="false"/>
        <property name="whenConstraint">
            <list>
                <bean parent="whenConstraint">
                    <property name="values">
                        <list>
                            <value>DRAFT</value>
                        </list>
                    </property>
                    <property name="constraint">
                        <bean parent="baseKualiOrgIdRepeating">
                            <property name="caseConstraint">
                                <bean parent="caseConstraint">
                                    <property name="operator" value="EQUALS" />
                                    <property name="fieldPath" value="proposal/workflowNode" />
                                    <property name="whenConstraint">
                                        <list>
                                            <bean parent="whenConstraint">
                                                <property name="values">
                                                    <list>
                                                        <value>PreRoute</value>
                                                        <value>Document Organization Review</value>
                                                        <value>Publication Decision Review</value>
                                                        <value>Department Review</value>
                                                        <value>Division Review</value>
                                                        <value>College Review</value>
                                                        <value>Senate Review</value>
                                                        <value>Publication Review</value>
                                                    </list>
                                                </property>
                                            </bean>
                                        </list>
                                    </property>
                                </bean>
                            </list>
                        </property>
                    </bean>
                </list>
            </property>
        </bean>
    </list>
</property>
```

```
</bean>
</property>
</bean>
<bean parent="whenConstraint">
<property name="values">
<list>
<value>APPROVED</value>
<value>ACTIVE</value>
<value>RETIRED</value>
</list>
</property>
<property name="constraint">
<bean parent="baseKualiOrgIdRepeatingRequired" />
</property>
</bean>
</list>
</property>
```

```
</bean>  
</property>
```

## (CM 2.0) 6.3 Document Types

To define the business process and where approvals may be needed for a certain workflow document, the Kuali Workflow system uses a *Document Type*. The document type defines many elements of the routing configuration, including the process order in the *routePath* element and the labeling that will be used in the Workflow Action List for each type.

In KS Curriculum Management, the Workflow Document Type Name matches the KS *Proposal Type*. For example, for a Credit Course Creation proposal, the Proposal Type and the Workflow Document Type Name are both "kuali.proposal.type.course.create". There are some Workflow Document Types, such as CluCreditCourseParentDocument, that are only used as Workflow *parent document types*. Parent document types allow common elements to be inherited by other Workflow Document Types. This is especially useful for elements that are shared across all functions that can be performed against a single KS Curriculum Management object. For example, the CluCreditCourseParentDocument contains workflow elements that are used when a Credit Course is either created or modified.

The example below shows part of the XML that defines the document type for a course creation. The course creation inherits characteristics from the parent CluCreditCourseParentDocument document type.

```

<documentType>
    <name>kuali.proposal.type.course.create</name>
    <parent>CluCreditCourseParentDocument</parent>
    <description>Credit Course Proposal</description>
    <label>Credit Course Proposal</label>
    <postProcessorName>
        org.kuali.student.lum.workflow.CoursePostProcessorBase
    </postProcessorName>
    <active>true</active>
    <routingVersion>2</routingVersion>
    <routePaths>
        <routePath>
            <start name="PreRoute" nextNode="Department Review" />
            <role name="Department Review" nextNode="Division Review" />
            <role name="Division Review" nextNode="College Review" />
            <role name="College Review" nextNode="Senate Review" />
            <role name="Senate Review" nextNode="Publication Review" />
            <role name="Publication Review" />
        </routePath>
    </routePaths>
    <routeNodes>
        <start name="PreRoute">
            <activationType>P</activationType>
            <mandatoryRoute>false</mandatoryRoute>
            <finalApproval>false</finalApproval>
        </start>
        <role name="Department Review">
            <activationType>P</activationType>
            <qualifierResolverClass>
                org.kuali.student.lum.workflow.qualifierresolver.CocOrgTypeQualifierResolver
                </qualifierResolverClass>
                <organizationTypeCode>kuali.org.Department</organizationTypeCode>
                <organizationIdQualifierKey>departmentId</organizationIdQualifierKey>
            </role>
            ...
            <role name="Senate Review">
                <activationType>P</activationType>
                <qualifierResolverClass>
                    org.kuali.student.lum.workflow.qualifierresolver.StaticOrganizationQualifierResolver
                    </qualifierResolverClass>
                    <organizationId>141</organizationId>
                </role>
                <role name="Publication Review">
                    <activationType>P</activationType>
                    <qualifierResolverClass>
                        org.kuali.student.lum.workflow.qualifierresolver.StaticOrganizationQualifierResolver
                        </qualifierResolverClass>
                        <organizationId>176</organizationId>
                    </role>
                </routeNodes>
            </documentType>

```

## (CM 2.0) 6.4 Route Nodes

Each ‘stop point’ where workflow action requests can be generated is referred to as a *route node* (or simply *node*). The `<routePath>` xml element in the document type defines the order in which the nodes are processed. The `<routeNodes>` xml element is a list of the definitions of each route node. For specifics about route node configuration please see the Rice documentation at <http://site.kuali.org/rice/2.2.0/reference/html/portal.html>.

All built-in node configurations in KS Curriculum Management are meant to be used with the Kuali Identity Management (KIM) Responsibility routing method (see [6.4.4 Kuali Identity Management Constructs Used by Workflow](#)). In the document type xml configuration, the node tag of `<role>` is used so that the workflow engine instantiates and processes the node as an Identity Management-related node. Nodes using Identity Management routing must have the `<qualifierResolverClass>` xml element defined. This class uses the document content xml and the workflow document data to define and return role qualifications which are used to qualify roles to find relevant role members.

Another method of routing that can be used is the Workflow *Rule Based Routing* where xml can be ingested into Workflow to facilitate various routing needs. Configuration for the Rule Based method can be found in the RICE documentation site as well.

### (CM 2.0) 6.4.1 Organization Routing Node Configurations

The built-in configuration for almost all document types in KS Curriculum Management are to be used with Organization-based routing. KS Curriculum Management has its own internal Organization Management and routing is set up to use that system. For example, workflow will route approval requests to organization members who hold the position of *Chair* while routing only FYI requests to members whose position is *committee member*.

Some configurations of these organization-based route nodes in KS Curriculum Management documents are customizations made specifically for KS Curriculum Management. You can use these configurations to facilitate individual routing needs, assuming that the KS Curriculum Management Organization structure is similar to that which is set by default in the KS Curriculum Management codebase. If the Organization structure is unlike the shipped version, you will need to make changes to the Workflow classes.

Follow these steps to add a new route node to the existing Workflow document type configuration:

1. Add the new data to the Organization Management system inside KS Curriculum Management. This includes the new Organization Type and new Organizations based on that type. If the Organization Management system has been set to the requirements of the implementing institution then no additional changes need to be made there.
2. Add a new node to the document type xml
  - a. Add the node to proper location in the `<routePath>` element.
  - b. Add the `<role>` node to the `<routeNodes>` element. See the configuration options immediately below.
3. Add a new Responsibility.

### (CM 2.0) 6.4.2 Qualifier Resolver Classes

The following qualifier resolver classes are delivered with KS Curriculum Management:

- Committee on Curricula Organization Type Node Configuration
- Static Organization Node Configuration
- Committee on Curricula Organization Node Configuration

The example configuration, attributes and role qualifier function for each of these nodes are provided in this section.

#### (CM 2.0) 6.4.2.1 Committee on Curricula Organization Type node Configuration (Org Tree)

Use this route node configuration to find the organizational hierarchy (parents, parents of parents, etc.) for the organization set in the document content xml if it is a certain Organization Type.

The following is an example configuration for the Curricula Organization Type.

```
<role name="College Review">
    <activationType>P</activationType>
    <qualifierResolverClass>

        org.kuali.student.lum.workflow.qualifierresolver.CocOrgTypeQualifierResolver
    </qualifierResolverClass>
    <organizationTypeCode>kuali.org.College</organizationTypeCode>
    <organizationIdQualifierKey>collegeId</organizationIdQualifierKey>
    <organizationShortNameQualifierKey>college</organizationShortNameQualifierKey>
</role>
```

#### (CM 2.0) 6.4.2.1.1 Attributes

These attributes can be used with this node configuration:

Attribute Name	Required	Attribute Value	Default Value
qualifierResolverClass	Yes	org.kuali.student.lum.workflow.qualifierresolver.CocOrgTypeQualifierResolver	N/A
organizationTypeCode	Yes	The organization type code from the Kuali Student Organization Management system that will be used to routed to.	N/A
organizationIdQualifierKey	No	Any text value	orgId
useNonDerivedRoles	No	True or false	true
organizationShortNameQualifierKey	Dependent	Value that matches KIM role qualifier key for Organization Short Name	N/A
organizationIdDocumentContentKey	Dependent	Value that matches KIM role qualifier key for Organization ID	N/A

Both the <qualifierResolverClass> and the <organizationTypeCode> attributes must be set in this configuration. The <organizationTypeCode> must be set to a valid Organization Type from the Organization Management system.

The <organizationIdQualifierKey> allows institutions to override the xml tag that will be used to find the organization id(s) in the document content. If this attribute is not set, the system will look for one or more <orgId> xml tags.

In this node configuration, the use of ad-hoc roles is simplified by some of the optional attributes. Ad-hoc roles allow an institution to add members to the normal workflow process without adding them to positions in the Kuali Student Organization Management system. The <useNonDerivedRoles> attribute can be used to turn off the ad-hoc role qualifications. If the <useNonDerivedRoles> attribute is set to false then the <organizationShortNameQualifierKey> and <organizationIdDocumentContentKey> attributes will not be required and will be used only if they are set. If the <useNonDerivedRoles> attribute is set to true (or is not set in the node configuration at all) then the <organizationShortNameQualifierKey> and <organizationIdDocumentContentKey> attributes are required. The values may or may not be used by ad-hoc roles related to the Kuali Identity Management system but are required in the document type xml.

#### (CM 2.0) 6.4.2.1.2 Role Qualifier Function

The qualifier resolver class will first find all ancestor organizations related to the organization(s) specified in the Workflow document content xml. The "kuali.org.hierarchy.Curriculum" hierarchy type is used when searching for ancestor organizations and is hardcoded in the qualifier resolver class. Only organizations that are of the type defined by the <organizationTypeCode> attribute are returned in the resulting list.

When the qualifier resolver has obtained the list of ancestor organizations of the desired type, it then uses each match and try to find an organization that is related to the matched organization that has a relation of type "kuali.org.CurriculumParent" and the organization type of "kuali.org.COC". Both values are hardcoded in the qualifier resolver class

If any organizations are found, the *Organization ID* and *Organization Short Name* values are returned from the qualifier resolver class using the role qualifier keys "orgId" and "org" respectively. If the <organizationShortNameQualifierKey> and <organizationIdDocumentContentKey> attributes are set in the document type xml, those values also are returned as role qualifier keys when returning the *Organization Short Name* and *Organization ID*.

#### (CM 2.0) 6.4.2.2 Static Organization Node Configuration

This route node configuration is used when the node in question should use a specific organization from the Kuali Student Organization Management system.

The following is an example configuration for the Static Organization Type.

```

<role name="Senate Review">
    <activationType>P</activationType>
    <qualifierResolverClass>
        org.kuali.student.lum.workflow.qualifierresolver.StaticOrganizationQualifierResolver
    </qualifierResolverClass>
    <organizationId>141</organizationId>
</role>

```

#### (CM 2.0) 6.4.2.2.1 Attributes

These attributes must be defined for this node configuration:

Attribute Name	Attribute Value
qualifierResolverClass	org.kuali.student.lum.workflow.qualifierresolver.StaticOrganizationQualifierResolver
organizationId	The organization id from the Kuali Student Organization Management system that this qualifier resolver should return.

Both the <qualifierResolverClass> and the <organizationTypeCode> attributes must be set in this configuration. The <organizationId> must be set to a valid Organization from the Organization Management system.

#### (CM 2.0) 6.4.2.2.2 Role Qualifier Function

The qualifier resolver class first ensures that the organization id set in the document type xml is a valid organization. The system then fetches the *Organization Short Name* from the KS Organization Management system. The qualifier resolver class returns both the *Organization ID* and *Organization Short Name* using the role qualifier keys "orgId" and "org" respectively.

#### (CM 2.0) 6.4.2.3 Committee on Curricula Organization Node Configuration (Specific Org)

This route node configuration is used when the node in question should use a specific organization from the Kuali Student Organization Management system.

The following is an example of the Committee on Curricula Organization Node

```
<role name="Document Organization Review">
    <activationType>P</activationType>
    <qualifierResolverClass>
        org.kuali.student.lum.workflow.qualifierresolver.CocOrganizationQualifierResolver
    </qualifierResolverClass>
    <organizationIdQualifierKey>orgId</organizationIdQualifierKey>
    <organizationShortNameQualifierKey>org</organizationShortNameQualifierKey>
</role>
```

#### (CM 2.0) 6.4.2.3.1 Attributes

These attributes can be used with this node configuration:

Attribute Name	Required	Attribute Value	Default Value
qualifierResolverClass	Yes	org.kuali.student.lum.workflow.qualifierresolver.CocOrganizationQualifierResolver	N/A
organizationIdQualifierKey	No	Any text value	orgId
useNonDerivedRoles	No	True or false	true
organizationShortNameQualifierKey	Dependent	Value that matches KIM role qualifier key for Organization Short Name	N/A
organizationIdDocumentContentKey	Dependent	Value that matches KIM role qualifier key for Organization ID	N/A

The <qualifierResolverClass> attribute must be set to "org.kuali.student.lum.workflow.qualifierresolver.CocOrganizationQualifierResolver" in this configuration.

The <organizationIdQualifierKey> allows institutions to override the xml tag that will be used to find the organization id(s) in the document content. If this attribute is not set, the system will look for one or more <orgId> xml tags.

In this node configuration the use of ad-hoc roles is made easier by some of the optional attributes. Ad-hoc roles allow an institution to add members to the normal workflow process without having to add them to positions in the Kuali Student Organization Management system. The <useNonDerivedRoles> attribute can be used to turn off the ad-hoc role qualifications. If the <useNonDerivedRoles> attribute is set to false then the <organizationShortNameQualifierKey> and <organizationIdDocumentContentKey> attributes will not be required and will be used only if they are set. If the <useNonDerivedRoles> attribute is set to true (or is not set in the node configuration at all) then the

<organizationShortNameQualifierKey> and <organizationIdDocumentContentKey> attributes are required. The values may or may not be used by ad-hoc roles related to the Kuali Identity Management system, but are required in the document type xml.

#### (CM 2.0) 6.4.2.3.2 Role Qualifier Function

The qualifier resolver class finds all the organization ID(s) set in the Workflow document content xml. For each organization ID found, it finds an associated organization that is related to each will then use each of those organization IDs to try to find an associated organization that is related with the relation of type "kuali.org.CurriculumParent" and that has the organization type "kuali.org.COC". Both values are hardcoded in the qualifier resolver class.

If any organizations are found using the above process then the *Organization ID* and *Organization Short Name* values will be returned from the qualifier resolver class using the role qualifier keys "orgId" and "org" respectively. If the <organizationShortNameQualifierKey> and <organizationIdDocumentContentKey> attributes are set in the document type xml then those values will also be returned as role qualifier keys when returning the *Organization Short Name* and *Organization ID*.

### (CM 2.0) 6.4.3 Workflow Post Processing Configuration

Workflow uses an internal, asynchronous engine process that calculates the routing of a particular document instance. There is no user interaction within the process when it is run on the application server. Workflow allows the Workflow document type to have a *Post Processor Class* specified which allows application to be notified of actions that the engine is taking or processing. The Post Processor Class must be made available to the application that contains the Workflow engine processing. In KS Curriculum Management, the standalone RICE server is the application that contains the engine for the various KS modules.

A post processor class allows for the following 'plug-in' points within the engine processing:

Post Processor Class Method	Function
beforeProcess()	Runs before the engine begins asynchronous engine processing dealing with routing calculations.
afterProcess()	Runs after the engine completes the asynchronous engine processing dealing with routing calculations.
doActionTaken()	Runs when a Workflow action is being performed. The method has data that can differentiate the action being taken.
doRouteStatusChange()	Runs when the status of the Workflow document instance is being changed. The method allows access to the previous status and the new status.
doRouteLevelChange()	Runs when the Workflow document instance is moving from one route node to another. The method has data for what node name the document is leaving and to what node name the document is moving.
doDeleteRouteHeader()	Runs when Workflow is removing a document instance from the system.
getDocumentIdsToLock()	Used to allow a document to lock other documents while post processing occurs. Useful if the document instance may update other document instances as a result of post processing.

Post processing classes are provided by default for existing document types delivered with KS Curriculum Management. You can view the class by looking at the document type inquiry page in the RICE portal screens. By default KS Curriculum Management does not use the post processing methods beforeProcess(), afterProcess(), doDeleteRouteHeader() or getDocumentIdsToLock().

If an institution wants to override a bootstrapped post processor class for a specific document type or if they are creating a new KS Curriculum Management document type (that deals with a proposal), there is a few base post processor classes that can be used to plug into various KS Curriculum Management processes like *Collaborators* with Ad-Hoc Permissions or built in Proposal Status and Course status changes.

For more information about Post Processors, see the RICE documentation at <http://site.kuali.org/rice/2.2.0/reference/html/portal.html>.

#### (CM 2.0) 6.4.3.1 KS Curriculum Management Post Processor Base

The topmost post processor base class that can be used for KS Curriculum Management documents is *org.kuali.student.lum.workflow.KualiStudentPostProcessorBase*. This post processor class performs the following operations:

1. This class changes the Proposal Status based on the changing Workflow status. All statuses should be the same between Workflow and the Proposal with the exception of the following.
  - a. Workflow "DISAPPROVED" status will set the Proposal to "Rejected" status
  - b. Workflow "PROCESSED" status will set the Proposal to "Approved" status
2. If an action is taken by an invited Collaborator, this post processor will remove that person's ability to edit the document if they had been given that permission.
3. If the document instance in Workflow is transitioning out of the start node (it must be named "PreRoute"), then this post processor will

remove the ability to edit the proposal from all Collaborators.

Some custom KS Curriculum Management methods in this post processor class can be overridden to simplify implementation. Some of the most useful are as follows (this is not a complete list):

1. processCustomSaveActionTaken(ActionTakenEvent, ActionTakenValue) – Allows for a plug-in point for when a 'save' action is being taken but also allows default operations to run properly. The ProposallInfo object may not be available if the proposal has never been saved before.
2. processCustomSaveActionTaken(ActionTakenEvent, ActionTakenValue, ProposallInfo) – Allows for a plug-in point for when a non-save action is being taken but also allows default operations to run properly.
3. processWithdrawActionTaken(ActionTakenEvent, ProposallInfo) – Allows for a plug-in point for when a proposal is being withdrawn but also allows default operations to run properly.
4. processCustomRouteLevelChange(DocumentRouteLevelChange, ProposallInfo) – Allows for a plug-in point that runs when the document is moving from one node to another but also allows default operations to run properly.
5. processCustomRouteStatusSavedStatusChange(DocumentRouteStatusChange) – Allows for a plug-in point that runs when the document is changing from Workflow status "INITIATED" to Workflow status "SAVED". It allows for the default operations to be performed as normal.
6. processCustomRouteStatusChange(DocumentRouteStatusChange, ProposallInfo) - Allows for a plug-in point that runs when the document is changing from one Workflow status to another but not the case where the document is changing from "INITIATED" to "SAVED" status. This method allows for the default operations to be performed as normal.

#### **(CM 2.0) 6.4.3.2 KS Curriculum Management Course Post Processor Base**

There is a post processor base class used by the Course document types (Create and Modify). The class is `org.kuali.student.lum.workflow.CoursePostProcessorBase`. This class performs all the standard operations of the `org.kuali.student.lum.workflow.KualiStudentPostProcessorBase` class . This class will also change the Course Status based on the changing Workflow status. The Course status will be set in the following ways based on the new Workflow status:

1. Workflow "SAVED" status will set Course "Draft" status
2. Workflow "ENROTUE" status will set Course "Submitted" status
3. Workflow "PROCESSED" status will set Course "Approved" status

Some custom KS Curriculum Management methods in this post processor class can be overridden to ease implementation. All the same methods apply in this class that apply in the `org.kuali.student.lum.workflow.KualiStudentPostProcessorBase` class. One of the methods that this class also has is `preProcessCluSave(IDocumentEvent, CluInfo)`. This method allows any extension of this base class to perform custom actions prior to the CLU object being saved to the database. The method should return 'true' if the implementation requires a save to occur or 'false' if a save of the CLU object does not need to occur. The `IDocumentEvent` object can be used to identify whether the save is resulting because of a Workflow status change event or an action taken event. An implementing institution can check to see if the `IDocumentEvent` object is an instance of these classes:

1. Workflow Action Taken - `org.kuali.rice.kew.postprocessor.ActionTakenEvent`
2. Workflow Status Change - `org.kuali.rice.kew.postprocessor.DocumentRouteStatusChange`

#### **(CM 2.0) 6.4.4 Kuali Identity Management Constructs Used by Workflow**

When using the Kuali Identity Management (KIM) Responsibility method of routing, the KIM constructs needed by Workflow are *Responsibilities*, *Roles*, and *Principals*. These constructs are combined by the Workflow system to retrieve a list of principals and data that pertains to how those principals should be routed to. That data is used to create the standard Workflow *Action Request* objects for each document instance and each request then relates to the *Action Item* objects that show in each individual principal's Action List.

##### **(CM 2.0) 6.4.4.1 Principals**

Principals in the KIM system are basic representations of individuals or systems (usually ones that can authenticate with the system). You can find much more information on how to configure your existing Identity Management systems with KIM in the RICE documentation online. For this document we will assume valid principals are set up in the system.

##### **(CM 2.0) 6.4.4.2 Roles**

In KIM, a role is a method of grouping common principals and allowing each principal assigned to the role to have certain qualifications. An example might be a role of "Campus Manager" where there can be individual principals assigned and each principal can be *qualified* by data such as the campus that the principal manages. In this way the system can retrieve the "Campus Manager" user for the "North" campus, rather than pulling in all "Campus Manager" principals.

KS Curriculum Management also has a number of Derived Roles (also called Application Roles). Derived Roles do not have explicit principals assigned within the KIM system. This is because a Derived Role will derive the principals from an external system's data or services. This is how KS Curriculum Management gets principals associated with the Organization Management system. Principals are assigned positions within the KS Organization Management system and then KIM is able to have derived roles to retrieve that data.

Some Organization Routing specific Derived Roles are as follows:

Role Name	Qualifier Key (Required)	Description
-----------	--------------------------	-------------

Org Admin Reviewer	orgId (yes) org (no)	Role will use the orgId qualification to find the principals that hold the position of "kuali.org.PersonRelation.Chair" for that organization ID
Org Committee Reviewer	orgId (yes) org (no)	Role will use the orgId qualification to find the principals that hold any position that is not "kuali.org.PersonRelation.Chair" for that organization ID

KS Curriculum Management also includes examples of ad-hoc roles for organization routing. An example of that setup is the two KS-LUM namespaced roles "Department Admin Reviewer" and "Department Committee Reviewer". The "Department Admin Reviewer" role is a way to add a principal as a certain Organization's position of "kuali.org.PersonRelation.Chair" for purposes of routing without having to put them into the actual Organization system in KS Curriculum Management. Likewise the "Department Committee Reviewer" role is used in a similar way to allow a principal to be included in the routing of an Organization but not force the implementing institution to add that principal to the Organization Management system positions.

#### (CM 2.0) 6.4.4.3 Responsibilities

A Responsibility is the object in KIM that links the Workflow document type route node to a set of KIM roles that should be used to calculate which principals should receive Workflow requests.

In basic terms, the KIM Responsibility object is created with *Responsibility Details* which include the *documentTypeName* key and the *routeNodeName* key. These data elements allow Workflow to find the Responsibility objects that match its current document instance and the route node that the current document instance is using. Workflow will retrieve these Responsibilities by matching the *Workflow Document Type Name* and the *Route Node Name* values to the responsibility detail data elements. The document type hierarchy (parent document types) will be taken into account so that Responsibilities can be set up using parent document types if desired. For more information on default behavior of Responsibilities and how Workflow uses them see the RICE Documentation "KEW Technical Reference Guide" and more specifically the "Matching Routing Nodes to Responsibilities" page.

Responsibilities have a *Responsibility Detail* value for the key "required". If the value is set to "true" and no principals are returned from the calculation of this Responsibility, the document will go into exception status. If the value is set to "false" and no principals are returned then the document will continue on the original route path.

Responsibilities also have a *Responsibility Detail* value for the key "actionDetailsAtRoleMemberLevel". This value is either 'true' or 'false' to indicate whether the Action Detail elements are stored at the role member level or role level. A value of 'true' indicates that each member of the role will have its own Action Detail data element while a value of 'false' will require that the role have an Action Detail element that then applies to all role members. The following fields are available in the Action Detail object:

Detail	Valid Values (DB Value)	Description
Action Type Code	APPROVE (A) ACKNOWLEDGE (K) FYI (F)	Indicates the type of action request that KEW should generate.
Force Action	TRUE (Y) FALSE (N)	If FALSE, KEW will check a document's route log to see if an earlier Action Taken can satisfy the request being generated for the current responsibility.
Action Policy Code	FIRST (F) ALL (A)	Signifies how the generated request will work in KEW in terms of when the responsibility is satisfied (whether one person needs to approve the document or all persons). "ALL" value is only allowed at the role member level for role members of type 'group'.
Priority Number	Numerals (NUMBER)	If the route node is using sequential routing, this priority will be used to decide in what order requests should be routed.

## (CM 2.0) 7. Configuring Rules

In KS Curriculum Management, the rules infrastructure is used to manage course restriction rules (pre-requisite, co-requisite, etc.) and program rules (also known as degree audit rules).

Each rule is composed of one or more "Requirement Component Types". A Requirement Component Type is something like "Must have completed *n* courses from this list". Kauli Student is delivered with a standard set of Requirement Component Types.

## Adding Additional Requirement Component Types

### 1. Creating a new Requirement Component Type:

First you will need to add the requirement component type to the KSST\_REQ\_COM\_TYPE table in the database. This consists of 5 fields that represent the requirement.

Example row of table KSST\_REQ\_COM\_TYPE

TYPE_KEY	TYPE_DESC	EFF_DT	EXPIR_DT	NAME
'kuali.reqComponent.type.program.credits.advisor'	'Must have <n> credits from courses approved by advisor'	SYSDATE	SYSDATE	'Minimum advisor approved credits'

### 2. Create links between the Result Component and its field types (variable parts of the rules).

Data must then be entered into the KSST\_RCTYP\_JN\_RCFLDTYP table.

Examples:

REQ_COMP_TYPE_ID	REQ_COMP_FIELD_TYPE_ID
'kuali.reqComponent.type.program.courses.theme'	'kuali.reqComponent.field.type.value.positive.integer'
'kuali.reqComponent.type.program.courses.advisor'	'kuali.reqComponent.field.type.value.freeform.text'

### 3. Create the Natural Language Translation for the result component/rule.

There are multiple contexts for the natural translation, which allow for different view of rule data.

#### KUALI.RULE.COMPOSITION:

This works with the UI to create editors for each rule.

Example rule composition:

```
<reqCompFieldType=kuali.reqComponent.field.type.program.cluSet.id;reqCompFieldLabel=Program(s)> no more than  
<reqCompFieldType=kuali.reqComponent.field.type.value.positive.integer;reqCompFieldLabel=Number of Courses>  
in  
<reqCompFieldType=kuali.reqComponent.field.type.org.id;reqCompFieldLabel=Department>  
in <reqCompFieldType=kuali.reqComponent.field.type.duration;reqCompFieldLabel=Duration Count>  
of type  
<reqCompFieldType=kuali.reqComponent.field.type.durationType.id;reqCompFieldLabel=Duration Type>
```

Notice how each field is represented in a tag format with the reqCompFieldType and reqCompFieldLabel set.

#### KUALI.RULE.EXAMPLE

This usage type is used in the UI to display an example of the rule:

```
Must have successfully completed no more than 2 courses from (MATH111, 140, 220, and STAT100)
```

## KUALI.RULE, KUALI.RULE.CATALOG, and KUALI.RULE.PREVIEW

These are used to display rules in different formats. There is logic that can be added to call java functions to display different translations depending on the field values.

```

Must have earned a minimum GPA of $gpa in
\#if($courseCluSet.getCluList().size() == 1)Must be concurrently enrolled
in#{else}Must be concurrently enrolled in all courses from#end
Must have successfully completed a minimum of $intValue
$NLHelper.getProperGrammar($intValue, "program") from
$programCluSet.getCluSetAsLongName()
$NLHelper.getProperGrammar($programCluSet.getCluList().size(), "program")
Must not have successfully completed #if($programCluSet.getCluList().size() > 1)any of
#end$programCluSet.getCluSetAsLongName()
$NLHelper.getProperGrammar($programCluSet.getCluList().size(), "program")

```

KSST\_REQ\_COM\_TYPE\_NL\_TMPL example:

ID	ATTR_NAME	ATTR_VALUE	LANGUAGE	NL_USAGE_TYPE_KEY	TEMPLATE
108	(null)	(null)	en	KUALI.RULE.PREVIEWS	Must have earned a minimum GPA of \$gpa in

#### 4. Linking the new Requirement Component Type to a Statement Type

Once a requirement component is complete, it will need to be linked to statement types to show up as a valid rule for different contexts. This is done in the KSST\_STMT\_TYP\_JN\_RC\_TYP table.

Example Row:

ID	STMT_TYPE_ID	REQ_COM_TYPE_ID
'1234567'	'kuali.statement.type.program.completion'	'kuali.reqComponent.type.program.credits.theme'

#### Example SQL statements for a rule type

Student must successfully complete <n> credits from <theme>

```

--Student must successfully complete <n> credits from <theme>
INSERT INTO KSST_REQ_COM_TYPE (TYPE_KEY, TYPE_DESC, EFF_DT, EXPIR_DT, NAME) VALUES
('kuali.reqComponent.type.program.credits.theme', 'Student must successfully complete
<n> credits from <theme>', SYSDATE, SYSDATE, 'Minimum theme approved credits')
/
insert into KSST_RCTYP_JN_RCFLDTYP (REQ_COMP_TYPE_ID,REQ_COMP_FIELD_TYPE_ID) values
('kuali.reqComponent.type.program.credits.theme',
'kuali.reqComponent.field.type.value.positive.integer')
/
insert into KSST_RCTYP_JN_RCFLDTYP (REQ_COMP_TYPE_ID,REQ_COMP_FIELD_TYPE_ID) values
('kuali.reqComponent.type.program.credits.theme',
'kuali.reqComponent.field.type.value.freeform.text')
/
insert into KSST_REQ_COM_TYPE_NL_TMPL (ID, ATTR_NAME, ATTR_VALUE, LANGUAGE,
NL_USAGE_TYPE_KEY, TEMPLATE, OWNER) values ('4006', '', '', 'en',
'KUALI.RULE.EXAMPLE', 'Student must successfully complete 35 credits in music theory
or analysis', 'kuali.reqComponent.type.program.credits.theme')
/
insert into KSST_REQ_COM_TYPE_NL_TMPL (ID, ATTR_NAME, ATTR_VALUE, LANGUAGE,
NL_USAGE_TYPE_KEY, TEMPLATE, OWNER) values ('4106', '', '', 'en',
'KUALI.RULE.COMPOSITION',
'<reqCompFieldType=kuali.reqComponent.field.type.value.positive.integer;reqCompFieldLa
bel=Minimum Number of Credits> in
<reqCompFieldType=kuali.reqComponent.field.type.value.freeform.text;reqCompFieldLabel=
Theme>', 'kuali.reqComponent.type.program.credits.theme')
/
insert into KSST_REQ_COM_TYPE_NL_TMPL (ID, ATTR_NAME, ATTR_VALUE, LANGUAGE,
NL_USAGE_TYPE_KEY, TEMPLATE, OWNER) values ('4206', '', '', 'en',
'KUALI.RULE.CATALOG', 'Student must successfully complete $intValue
$NLHelper.getProperGrammar($intValue, "credit") in
$fields.get("kuali.reqComponent.field.type.value.freeform.text")',
'kuali.reqComponent.type.program.credits.theme')
/
insert into KSST_REQ_COM_TYPE_NL_TMPL (ID, ATTR_NAME, ATTR_VALUE, LANGUAGE,
NL_USAGE_TYPE_KEY, TEMPLATE, OWNER) values ('4306', '', '', 'en',
'KUALI.RULE.PREVIEW', 'Student must successfully complete $intValue
$NLHelper.getProperGrammar($intValue, "credit") in
$fields.get("kuali.reqComponent.field.type.value.freeform.text")',
'kuali.reqComponent.type.program.credits.theme')
/
insert into KSST_REQ_COM_TYPE_NL_TMPL (ID, ATTR_NAME, ATTR_VALUE, LANGUAGE,
NL_USAGE_TYPE_KEY, TEMPLATE, OWNER) values ('4406', '', '', 'en', 'KUALI.RULE',
'Student must successfully complete $intValue $NLHelper.getProperGrammar($intValue,
"credit") in $fields.get("kuali.reqComponent.field.type.value.freeform.text")',
'kuali.reqComponent.type.program.credits.theme')
/
insert into KSST_STMT_TYP_JN_RC_TYP (ID, STMT_TYPE_ID, REQ_COM_TYPE_ID) values
('PC-406','kuali.statement.type.program.completion',
'kuali.reqComponent.type.program.credits.theme')
/

```

## 5. Configure the context

Once the Rule is added to the database via the SQL, some configuration needs to occur. This example shows adding a lookup for course prefixes reusing the existing subject area search:

Add the rule component type to the statement context:

statement-context.xml:

```
<bean id="contextRegistryReqComponent">
    <constructor-arg>
        <map>
            ...
            <entry key="kuali.reqComponent.type.program.courses.theme">
                <ref local="basicContext"/>
            </entry>
            ...
        </map>
    </constructor-arg>
</bean>
```

Add an entry to the contextRegistryReqComponent bean, with the key of the requirement component type you are adding, and a reference to a context. For most cases you will use the basicContext, which has access to basic lookups and widgets. If you need custom widgets or fields that require specific context, you might need to use additional contexts. As an example, the Course search fields require the luContext.

## 6. Configure the datadictionary and lookups

If your requirement component has fields that require lookups or special UI, you will need to configure the data dictionaries:

in ks-statement-dictionary-context.xml:

Add a where constraint that will map the field type to a specific validation constraint:

```

<bean id="reqCompFieldInfo.value-parent" abstract="true" parent="baseStringLong">
    <property name="name" value="value"/>
    <property name="caseConstraint">
        <bean parent="caseConstraint">
            <property name="operator" value="EQUALS"/>
            <property name="fieldPath" value="type"/>
            <property name="caseSensitive" value="false"/>
            <property name="whenConstraint">
                <list>
                    ...
                </list>
            </property>
            <bean parent="whenConstraint">
                <property name="values">
                    <list>
                        <value>kuali.reqComponent.field.type.course.prefix</value>
                    </list>
                </property>
                <property name="constraint">
                    <bean parent="reqCompFieldInfo.value.course.prefix"/>
                </property>
            </bean>
        </property>
        ...
    </list>
</property>
</bean>
</property>
</bean>

```

You can add the constraint to that file if a constraint does not yet exist:

```

<bean id="reqCompFieldInfo.value.course.prefix-parent" abstract="true"
parent="baseKualiCodeLoose" >
    <property name="lookupDefinition" ref="constraint.lookup.subjectCodes" />
    <property name="minOccurs" value="1" />
    <property name="maxLength" value="4"/>
</bean>
<bean id="reqCompFieldInfo.value.course.prefix"
parent="reqCompFieldInfo.value.course.prefix-parent" />

```

If your field has a lookup, you will need to configure the lookup in the proper lookup context e.g. lum-ui-lookup-context.xml

```

<bean id="ReqCompFieldInfo.value.course.prefix" parent="uilookupconfig-parent">
    <property name="path" value="reqCompFieldInfo.value" />
    <property name="type" value="kuali.reqComponent.field.type.course.prefix" />
    <property name="initialLookup" ref="kuali.subjectCode.search.subjectAreas" />
</bean>

```

The type property should match the name of your field type.

## (CM 2.0) 8. Authorization

### Definitions

These terms will be referred to throughout this document. Although these terms probably aren't very meaningful at this point, hopefully things will make more sense after walking through an authorization example.

**Principal:** Represents a KIM entity that can authenticate into a Kuali system.

**Role:** Representing a "function" that a role member may be able to do. Roles aggregate Permissions and Responsibilities. Roles have members that are Groups, Roles(nesting), or Principals.

#### Derived Role:

- Is a type of role, where the information is not captured with in KIM, but is part of the Application ie LUM, and needs to be reused with in permission
- These types of roles are NOT assigned to a Principal, they are "determined" at run time.
- These roles are linked to a Matching Service which then access the Application information and determine whether or not the principal id has access or not.

**Permission:** Defines a granular task that is non-workflow related. This answers the question: **Can** user 'jDoe' do 'foo'? Assigned to a role(s).

**Responsibility:** Defines a granular task that is workflow related. This answers the question: **Must** user 'jDoe' do 'foo'? Assigned to a role(s).

**KIMType Definition:** Defines the authorization criteria for elements of KIM. The criterion defines member attribute names and a **Matching service** which ties everything back to the java code. Both are optional.

#### Matching service:

- KIM will invoke the Matching service defined in a KIM Type.
- Since many things are KIM Types, multiple services may get invoked while looking for authorization.
- These services determine things like: What permissions "match" the name "create" which the permission attributes docType=Award? or Is the principal in the role FooBar?
- Ultimately, these matching services define authorization in KIM.
- These matching services are the link back to the Java code.
- There are existing services with in Kim, but custom ones are written within the Application ie KS, LUM
- This service must also be remotely accessible from KIM for KIM to work as a central installation - this is possible thru spring configuration

#### Attribute Definition:

- A Name/Value pair used to check authorization.
- These attributes are defined in the database as attribute definitions.
- They are matched via static values specified in the KIM database if using the default Service implementations.
- If not using the defaults you can provide your own matching service logic ignoring the attribute values stored in the database.

**Namespace:** A way to organize elements of KIM based on System, Subsystem, module, etc. This also helps prevent name clashes between elements of KIM since each element is assigned to a namespace.

**Template:** Permissions and Responsibilities have templates. These templates allow the reuse of permission/responsibility data within multiple permissions/responsibilities while specifying different attribute values.

## (CM 2.0) 8.1 Field Level Authorization

KS Curriculum Management can be configured to restrict or grant access to data at the field-level for all fields defined in the Data Orchestration Dictionary. By default every field has unrestricted access, which provides all users with full view and edit privileges. To revoke full access to a particular field, you must place a restriction on the field. Users can then be granted permissions to bypass these restrictions.

### (CM 2.0) 8.1.1 Restricting Field Access

To prevent users from having full access to a field, you must place restrictions on the field to trigger a permission check for a user's level of access. There are three types of access restrictions that can be placed on a field. These are controlled by setting the restriction property to `false` or the field definition within the dictionary configuration file. If not set, these properties all default to true.

**NOTE:** As of the publication of this document, only the `readOnly` and `mask` restrictions have been implemented.

Restriction	Description	Restriction Property	Values
Read-Only	This restriction can be used to prevent users from editing a field value in edit screens.	<code>readOnly</code>	<code>true</code> – Users cannot edit the field value. The field value will be displayed as read-only in any view. <code>true</code> - Users can edit field value.

Hidden (Non-Viewable)	This restriction can be used to prevent display of a field's value to the user.	hide	<b>true</b> - Users cannot view the value of the field. The field will not be visible in any view.  <b>false</b> - Users can view the value of the field.
Masked	This restriction can be used to mask the values of a field.	mask	<b>true</b> – Users cannot see the unmasked value of the field. The field value will be masked in any view. How the field is masked is defined with the following properties: <i>maskFormatter</i> --fully masks the field <i>value</i> . <i>partialMaskFormatter</i> --masks part of the field value (e.g., *****1234)  <b>true</b> - Users will see the original, unmasked value of the field.

### (CM 2.0) 8.1.1.1 Adding a Field Restriction

Follow these steps to add a restriction to a field.

**NOTE:** Changes to field restrictions will not take effect until the server is restarted.

1. Identify the appropriate service dictionary (eg. ks-courseInfo-dictionary-context.xml)
2. Identify the bean definition of the field or section you wish to restrict. For example:

```
<bean id="courseInfo.courseTitle-parent" abstract="true"
parent="cluIdentifierInfo.longName-parent">
```

3. Add desired property setting to enable the restriction (if not specified, the default value is *false*).  
To make the field read-only (user cannot edit):

```
<property name="readOnly" value="true" />
```

To hide the field:

```
<property name="hide" value="true" />
```

To mask the value of the field:

```
<property name="mask" value="true" />
```

### (CM 2.0) 8.1.2 Defining Field Permissions

Once you have restricted access to the field, you must create permissions to bypass the restriction to allow access to that field. Field-level permissions are defined and created in Kuali RICE using the KS-SYS: Field Access" permission template.

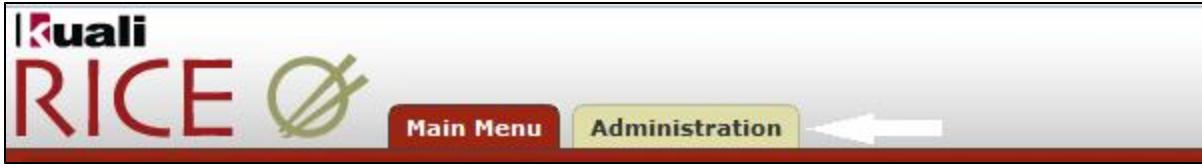
To set field permissions in Kuali RICE,

1. Enter Kuali RICE by clicking the **RICE** link:

Kuali Reference University



2. Click the **Administration** tab:



3. Click Permission:

A screenshot of the Kuali Rice administration interface, specifically the "Identity" section. The "Identity" tab is selected. On the left, there's a sidebar with a "Workflow" section containing links like Rule Attribute, Rule Template, XML Stylesheets, etc. In the center, under the "Identity" tab, there's a list of items: Person, Group, Role, **Permission** (which has a black arrow pointing to it), and Responsibility. On the right, there's a "Configuration" section with links like Parameter, Parameter Component, etc. At the top right, it says "Logged in User: admin".

4. Click create new:

A screenshot of the Kuali Rice administration interface, specifically the "Permission Lookup" page. It shows a list of permissions. On the right side, there's a "create new" button with a black arrow pointing to it. Below the "create new" button, it says "\* required field". At the top right, it says "Logged in User: admin".

5. Enter the permission information

The following permission details must be included:

6. You will need to define the permission with the following properties:

- **dtoName** – the name of the data object in the data dictionary (e.g., CreditCourseProposal)
- **dtoFieldKey** – the full path to the field in the data dictionary. The path is comprised of the "name" property of each element in the field's hierarchy joined by periods. For example, the *academicSubjectOrgs* (Curriculum Oversight) field is a child of *course*, so the full path to the field is *course.academicSubjectOrgs*.

If the field is a direct child of the data object, then the full path is simply the name of the field. For example, `course` is a direct child of `CreditCourseProposal`, so the full path is simply `course`.

<b>Permission</b>	<b>Doc Nbr:</b> 3024 <b>Status:</b> INITIATED <b>Initiator:</b> admin <b>Created:</b> 04:11 PM 04/13/20
-------------------	------------------------------------------------------------------------------------------------------------

\* required field

**Document Overview** ▼ hide

<b>* Description:</b> Curriculum Oversight Permission	<b>Explanation:</b> <input type="text"/>
<b>Org. Doc. #:</b> <input type="text"/>	

**Permission Info** ▼ hide

<b>New</b>	
<b>Permission Identifier:</b>	10000
<b>* Template Id:</b>	KS-SYS : Field Access
<b>* Permission Namespace:</b>	KS-SYS - Kuali Student System
<b>* Permission Name:</b>	Edit Curriculum Oversight
<b>Permission Description:</b>	Grants edit permission on the Curriculum Oversight field
<b>* Active Indicator:</b>	<input checked="" type="checkbox"/>

**Permission Details** ▼ hide

<b>New</b>	
	<b>Permission Details:</b> <code>dtoName=kuali.lu.type.CreditCourse dtoFieldKey=course.academicSubjectOrgs fieldAccessLevel=edit</code>

7. Click **Blanket Approve** to create the permission

### (CM 2.0) 8.1.3 Grant Field Permission

You must give field-level permission to a role in Kuali Rice to allow all users with that role to access restricted fields. Follow these steps to grant permission to a role in Kuali Rice:

1. Find or create a role to which you want to add the permission.
2. Add the field-level permission to the role.
3. Add the role to the user (if the user does not already have that role).

Once the permission is added to the role, all principals (users) assigned that role will have the field-level permission you've defined.

For details on managing roles and principals, see the [KIM User Guide](#).

## (CM 2.0) 8.2 KS Admin Screen Permissions and Role

### Requirement :

There are Admin screens and functions within LUM, for which admin authorization is required. This is implemented in the following way

### Implementation :

#### The Role:

- A new default role is created namely KS Admin Role, linked to the KS-SYS namespace
- This role can be assigned to any user that needs to be able to do the admin functions within LUM

#### The Permissions :

- A new permission template is created namely "KS Admin Screens" which is linked to the Kim Type 19 which is specific to the permissionPermissionTypeService service
- A new permission is created for each admin screen or function :
  - Home » Curriculum Management » Create Course --> On the popup screen there is a additional toggle, namely "Use curriculum review process for the course" which should be visible if the signed on user has the KS Admin Role. For this purpose a permission "KS Admin Screens" was created with Permission Details "screenComponent=useCurriculumReview"
    - screenComponent is a existing kim attribute which is just reused

- useCurriculumReview is the name of the toggle field to which the KS Admin Role must have access.
- Home » Curriculum Management » Find Course --> After selecting a course, a dropdown will appear filled with actions. The action "Propose Course Modification" will give the user who has the KS Admin Role access to different screen than a normal user. For this purpose a permission "KS Admin Course Workflow" was created with Permission Details "screenComponent=cluModifyItem"
  - cluModifyItem is the name of the drop list with the action values in that is different for the admin user than normal user.

**The Java Code:**

- On SecurityRpcGwtServlet.java a new method (checkAdminPermission) was implemented to call the Rice service to do a check for the permission created above.
- CurriculumHomeConfigurer.java was changed to call the SecurityRpcGwtServlet.checkAdminPermission sending the screenComponent to check the authorization, and to make the toggle field visible only if the method returns true which mean the user does have permission
- CourseWorkflowActionList.java was changed to call the SecurityRpcGwtServlet.checkAdminPermission sending the screenComponent to check the authorization, and to activate different screen for admin only if the method returns true which mean the user does have permission

**NOTE:**

These screens or function that is specific to the admin permission has to be setup in code and in data.

## (CM 2.0) 8.3 Creating Derived Role

**Derived Role:**

- Is a type of role, where the information is not captured with in KIM, but is part of the Application ie LUM, and needs to be reused with in permission
- These types of roles are NOT assigned to a Principal, they are "determined" at run time.
- These roles are linked to a Matching Service which then access the Application information and determine whether or not the principal id has access or not.

**Example:**

**Use Case :**

KS-LUM : Once final approval has been granted on a Course Proposal, any authenticated faculty or staff member can View the proposal.

**Things to note out of the use case:**

1. It is linked to "final approval", which is some or other attribute that we need to check
2. "faculty or staff" is affiliations that is captured with in KIM that we need to access - this can be seen as Application information, which mean we'll have to create a derived role

For the purpose of this example, we'll explain how to implement the Derived role.

### 1. Creating the KIM Type Definition - KRIM\_TYP\_T

- This will be used to determine which Matching Service we want to use to determine if the principal id is of certain affiliations. There are some predefined ones, for our purpose we'll extend one of these existing KIM services "KimDerivedRoleTypeServiceBase"
- **Important NOTE:** the service name specified here, MUST be used as the bean id when it is specified in the spring configuration later on, as this is the way that the KIM data links up with the java code.

```
INSERT INTO KRIM_TYP_T (ACTV_IND,KIM_TYP_ID,NM,NMSPC_CD,OBJ_ID,SRVC_NM,VER_NBR)
VALUES ('Y','3001','Derived
Role:Affiliation','KS-SYS','4d19e0c0-34fb-48a9-942e-8a1771c9d4c0','ksAffiliationService',1)
```

### 2. Creating the Matching service above in code.

- Within the project ks-lum-rice, package org.kuali.student.lum.kim.role.type create the following class
- Override the method hasApplicationRole - write the code here that will check if the principal id has the required affiliations
- the list of includedAffiliationTypes will be specified in the spring bean configuration

```

public class AffiliationDerivedRoleTypeServiceImpl extends
    KimDerivedRoleTypeServiceBase {

    private List<String> includedAffiliationTypes = null;

    @Override
    public boolean hasApplicationRole(String principalId,
        List<String> groupIds, String namespaceCode, String roleName,
        AttributeSet qualification) {
        Person signedOnPerson =
            KIMServiceLocator.getPersonService().getPerson(principalId);

        // check to see if principalID is a staff member or faculty member
        for (String affiliationType : includedAffiliationTypes) {
            if (signedOnPerson.hasAffiliationOfType(affiliationType)) {
                return true;
            }
        }
        return false;
    }

    public List<String> getIncludedAffiliationTypes() {
        return includedAffiliationTypes;
    }

    public void setIncludedAffiliationTypes(List<String> includedAffiliationTypes) {
        this.includedAffiliationTypes = includedAffiliationTypes;
    }
}

```

### 3. Specifying the Matching service in the spring bean config, so that it would be accessible from KIM instance.

- This service must also be remotely accessible from KIM for KIM to work as a central installation (ie Rice central installation used by KS and KFS for example) - this is possible thru spring configuration
- Add the service as a bean definition in the StudentStandaloneSpringBeans.xml for ks-web - to be accessible within LUM and within in ks-rice so that it will be accessible from the KIM Central installation
- Also specify the affiliation types that is valid
- **Important NOTE:** the bean id MUST correspond to the service name that was specified on the Kim Type

```

<!-- Affiliation Derived role (anyone that's either a Staff or Faculty affiliation)
-->
<bean id="ksAffiliationService"
>
<property name="includedAffiliationTypes">
<!-- See KimConstants.PersonAffiliationTypes -->
<list>
<value>STAFF</value>
<value>FACULTY</value>
</list>
</property>
</bean>

```

### Create the derived role

- We have to create a derived role, seen that we have some application specific data that we want to check
- This must be linked to the Kim Type that was created that specified the matching service name

```
insert into KRIM_ROLE_T (ROLE_ID, OBJ_ID, VER_NBR, ROLE_NM, NMSPC_CD, DESC_TXT,
KIM_TYP_ID, ACTV_IND, LAST_UPDT_DT)
values ('7000', 'c6699d07-33a3-4cf4-b1b6-030a3ca6a05d', '1', 'Derived Role :
Affiliation', 'KS-SYS', 'Derived Role : Affiliation', '3001', 'Y', null)
```

#### 4. Link the permission created earlier to the derived role

- For our use case we wanted to only apply the check on workflow docs that is in final status, for which we created the permission, so we have to link the permission to the derived role.

```
INSERT INTO KRIM_ROLE_PERM_T
(ACTV_IND,OBJ_ID,PERM_ID,ROLE_ID,ROLE_PERM_ID,VER_NBR)
VALUES ('Y','0bd38c98-bffa-4a41-b804-07e9b8a06642','3200','7000','6000',1)
```

Now test your derived role :

Remember, we created a derived role, which means it is not allocated to a principal, but will be determined at run time. SO if you run the LUM Application and go to the Rice Administration screen, you'll be able to see your role, but it will not have any members.

You can now run the application and test the derived role.

## (CM 2.0) 9. Authentication

KS Curriculum Management employs Spring Security as its core authentication framework and can be integrated into your Single Sign On (SSO) security system. If you have a Central Authentication Service (CAS) server at your institution, this can be done easily by following the instructions in the [KS Curriculum Management 2.0 Implementation and Deployment Guide](#).

**NOTE:** If you deploy the standalone version of Kuali Student and do not integrate Kuali Student and RICE with a Single Sign On (SSO) security system, the user will be prompted to login to RICE when performing actions that access RICE, such as viewing the Action List.

### (CM 2.0) 9.1 Spring Security

KS Curriculum Management uses Spring Security 3.1.0.RELEASE. Visit <http://projects.spring.io/spring-security/> to view the Spring Security 3.1.x Reference Manual to better understand Spring Security.

The standard KS Curriculum Management configuration uses Spring Security's Form Authentication Mechanism to provide a login form to the user. The Form Authentication Mechanism is described in detail in Chapter 11 of the Spring Security Reference Manual cited above.

#### (CM 2.0) 9.1.1 Overrides to Support Kuali Identity Management (KIM)

KS Curriculum Management uses Kuali Identity Management (KIM) for authorization. The following three overrides are employed to allow Spring Security to work with KIM. Depending on what SSO setup you have, you may need to implement some or all of these overrides.

- **UserWithId** (UserDetails) : just an extended User object with an additional userId property.
- **KSRiceDefaultUserDetailsService** (UserDetailsService) : uses the KIM identity service to get a KimPrincipal from the supplied login credentials. Using the KimPrincipal it builds and returns a UserWithId.
- **KimAuthenticationProvider** (AuthenticationProvider) : an authentication provider that calls KSRiceDefaultUserDetailsService and returns the UserWithId which is eventually stored in the SecurityContext.

## (CM 2.0) 9.2 Integrating Other Authentication Systems with KS Curriculum Management

You can integrate KS Curriculum Management with your institution's authentication system by providing the necessary hooks to Spring Security. Consult the [Spring Security 2.0.x Reference Manual](#) for details for instructions.

The exception to the instructions is that a UserWithId is returned from the UserDetailsService. If, instead, a custom UserDetails object needs to be

returned, the user name contained in this object must be the principal ID used for authorization checks through out KS.

## (CM 2.0) 9.3 Authentication with KS Curriculum Management and Rice

If have configured RICE with your institution's SSO, you need only configure KS Curriculum Management with your SSO as described in above (9.2. Integrating other Authentication Systems with ).

If RICE is not configured with your SSO, there is a sample configuration of RICE (with Spring Security), KS Standalone and CAS in the KS-RICE module. The following are the relevant parameters in ks-standalone-config.xml:

- *ks.ignore.rice.login*
  - Set to **true** so that KS Curriculum Management does not attempt to call RICE identity service to authenticate.
- *ks.rice.url*
  - Set to the full URL of the RICE webapp context to allow retrieval of the action list.

## (CM 2.0) 9.4 Configuring CAS with KS Curriculum Management Embedded

There are several things you'll need to do to use CAS in Curriculum Management Embedded mode. The broad overview is:

1. Create an institutional configuration project
2. Put some custom configuration files into that project
3. Build a war file from that project.
4. Alter your ...config.xml files to use CAS.

### Institutional Configuration

The guide on [setting up an institutional configuration project](#) is the place to start. For embedded mode, you'll want to copy the ks-with-rice-embedded deployment *and* the ks-rice-standalone deployment, rather than the ks-with-rice-bundled. Go through the other steps listed (such as editing the pom.xml file) to get the project specific to your site.

The current releases are at the following locations:

- <https://svn.kuali.org/repos/student/enrollment/ks-deployments/tags/ks-deployments-2.0.1-cm/ks-web/ks-with-rice-embedded/>
- <https://svn.kuali.org/repos/student/enrollment/ks-deployments/tags/ks-deployments-2.0.1-cm/ks-web/ks-rice-standalone/>

Copy those into separate directories with appropriate names (such as edu-ks-cm and edu-ks-rice, where edu is your institution).

After that, you'll need to create a file in the /src/main/resources/ directory of each of your institutional projects. Name it something like edu-cm-embedded-security.xml. You can create it by copying [the attached file](#), renaming it, and changing the comments to match your institution.

Next, edit the web.xml file in /src/main/webapp/WEB-INF/ in your institutional project. Change the contextConfigLocation parameter to match your copy of the file.

```
<!-- KS Spring Security Config - Begin -->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:edu-cm-embedded-security.xml</param-value>
</context-param>
```

Finally, run mvn clean install, followed by mvn war:war in the root of each project to build a war file that you can deploy. That done, you can edit the configuration files, as described below, and deploy your war.

### Changing Configuration Files

To configure KS Curriculum Management embedded to use your CAS server for authentication, set the following properties:

```
<param name="ks.default.security.cas.serverAddress">CASLoginURL</param>
<param name="ks.authentication.context">classpath:ks-spring-security-cas.xml</param>
```

where **CASLoginURL** is the URL for your CAS server login (for example, \*<https://login.ks.edu/cas>\*)

If your CAS server is using the Proxy Mechanism or if you want to enable it, set the following property:

```
<param name="ks.default.security.cas.useCasProxyMechanism">true</param>
```

**Note:** You must configure the server for SSL if you want to use the Proxy Mechanism.

## (CM 2.0) 10. Configuring Type Data

### (CM 2.0) 10.1 Degree Types

Degree Types are stored in the [Learning Result Catalog Service](#) as a result component. The type key for degree result components is *kuali.resultComponentType.degree*.

The Current Degree Types (database table: KSLR\_RESPCOMP) are:

ID	Name
kuali.resultComponent.degree.ba	Bachelor of Arts
kuali.resultComponent.degree.bmus	Bachelor of Music
kuali.resultComponent.degree.bsc	Bachelor of Science
kuali.resultComponent.degree.edd	Doctor of Education
kuali.resultComponent.degree.ma	Master of Arts
kuali.resultComponent.degree.mmus	Master of Music
kuali.resultComponent.degree.msc	Master of Science
kuali.resultComponent.degree.phd	Doctor of Philosophy

To add a new degree type, either use the *createResultComponent* operation of the LearningResultCatalogService or manually enter a new degree type into the KSLR\_RESPCOMP. The *Result Component Type* must be set to *kuali.resultComponentType.degree*.

Sample SOAP call to create degree type:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:lrc="http://student.kuali.org/wsdl/lrc">
    <soapenv:Header/>
    <soapenv:Body>
        <lrc:createResultComponent>

            <resultComponentTypeKey>kuali.resultComponentType.degree</resultComponentTypeKey>
            <resultComponentInfo type="kuali.resultComponentType.degree" state="Active" >
                <name>kuali.resultComponent.degree.jd</name>
                <desc>
                    <plain>Juris Doctor</plain>
                    <formatted>Juris Doctor</formatted>
                </desc>
            </resultComponentInfo>
        </lrc:createResultComponent>
    </soapenv:Body>
</soapenv:Envelope>
```

## (CM 2.0) 11. Misc Configuration

### (CM 2.0) 11.1 Copy Course & Copy Course Proposal - Fields to Copy

When performing the operations *copy existing course* and *copy existing course proposal*, the intention might be to not copy all of the properties. There is a setting in the spring configuration that can be overwritten for the *CopyCourseServiceImpl.java* class which will ignore properties so they are not copied. This will change the behavior for both copy course and copy course proposal.

In the spring config lum-gwt-context.xml override this bean and supply the properties that should not be copied:

```
<bean id="CopyCourseService"
  class="org.kuali.student.lum.lu.ui.course.server.gwt.CopyCourseServiceImpl">
  ...
  <property name="ignoreProperties">
    <list>
      <value>transcriptTitle</value>
      <value>unitsContentOwner</value>
      <value>unitsDeployment</value>
      <value>revenues</value>
      <value>expenditure</value>
    </list>
  </property>
</bean>
```

the values match our dictionary attribute names.



## **CM Services 2.0 Documentation**

(CM 2.0) Academic Time Period (ATP) Service .....	9
(CM 2.0) Academic Time Period Service Description and Assumptions .....	15
(CM 2.0) Storing and Querying Milestone Dates .....	16
(CM 2.0) Academic Time Period Service Entity Diagram .....	17
(CM 2.0) Academic Time Period Types and States .....	18
(CM 2.0) Academic Calendar Event Milestones .....	52
(CM 2.0) Admissions Milestones .....	52
(CM 2.0) Allowed ATP to ATP Types .....	53
(CM 2.0) ATP Relationships .....	54
(CM 2.0) ATP States .....	55
(CM 2.0) Billing Milestones .....	56
(CM 2.0) Calendar Types .....	57
(CM 2.0) Curriculum Milestones .....	57
(CM 2.0) Duration Types .....	59
(CM 2.0) Dynamic Attribute Types .....	61
(CM 2.0) Financial Aid Milestones .....	61
(CM 2.0) Graduation Milestones .....	62
(CM 2.0) Holiday Milestones .....	62
(CM 2.0) Key Date Milestones .....	67
(CM 2.0) Milestone States .....	73
(CM 2.0) Misc Milestones .....	74
(CM 2.0) Orientation Milestones .....	74
(CM 2.0) Program Related Types .....	75
(CM 2.0) Reporting Types .....	76
(CM 2.0) Scheduling Milestones .....	76
(CM 2.0) Seasonal Types .....	77
(CM 2.0) Seatpool Milestones .....	79
(CM 2.0) Term Types Grouping .....	80
(CM 2.0) Tuition Calculation Milestones .....	84
(CM 2.0) When Course Offered Types .....	85
(CM 2.0) Canonical Learning Unit (CLU) Service .....	86
(CM 2.0) Canonical Learning Unit Service Descriptions and Assumptions .....	87
(CM 2.0) Standardized Test Learning Unit Types and States .....	89
(CM 2.0) Standardized Test Learning Unit Types .....	89
(CM 2.0) Canonical Learning Unit Service Entity Diagram .....	89
(CM 2.0) Comment Service .....	90
(CM 2.0) Comment Service Description and Assumptions .....	92
(CM 2.0) Comment Service Types and States .....	93
(CM 2.0) Comment States .....	94
(CM 2.0) Comment Types .....	94
(CM 2.0) Comment Service Entity Diagram .....	95
(CM 2.0) Course Service .....	96
(CM 2.0) Course Service Description and Assumptions .....	97
(CM 2.0) Course Types and States .....	98
(CM 2.0) Course Types .....	101
(CM 2.0) Course Service Entity Diagram .....	104
(CM 2.0) Document Service .....	105
(CM 2.0) Document Service Description and Assumptions .....	106
(CM 2.0) Document Service Types and States .....	106
(CM 2.0) Document Service Entity Diagram .....	109
(CM 2.0) Enumeration Management Service .....	110
(CM 2.0) Enumeration Management Service Description and Assumptions .....	111
(CM 2.0) Enumeration Management Service Types and States .....	111
(CM 2.0) Enumeration Management Entity Model .....	112
(CM 2.0) Learning Objective (LO) Service .....	112
(CM 2.0) Learning Objective Service Description and Assumptions .....	113
(CM 2.0) Learning Objective Types and States .....	114
(CM 2.0) Learning Objective Service Entity Diagram .....	114
(CM 2.0) Learning Result Catalog (LRC) Service .....	115
(CM 2.0) Learning Result Catalog Change Log .....	116
(CM 2.0) Learning Result Catalog Service Description and Assumptions .....	118
(CM 2.0) Learning Result Catalog Types and States .....	119
(CM 2.0) Mapping from R1 Result Components .....	123
(CM 2.0) Result Scale Lifecycle Process States .....	124
(CM 2.0) Result Scale Types .....	125
(CM 2.0) Result Types and Structures from R1 .....	126
(CM 2.0) Result Value Group Types .....	134
(CM 2.0) Result Value Lifecycle Process States .....	134
(CM 2.0) Result Values Group Lifecycle Process States .....	135
(CM 2.0) Result Value Types .....	135
(CM 2.0) Learning Result Catalog Gordon Documentation .....	136
(CM 2.0) Learning Result Catalog Service Entity Diagram .....	137

(CM 2.0) Message Service .....	138
(CM 2.0) Message Service Description and Assumptions .....	139
(CM 2.0) Message Service Types and States .....	139
(CM 2.0) Message Service Entity Diagram .....	140
(CM 2.0) Organization Service .....	140
(CM 2.0) Organization Service Description and Assumptions .....	141
(CM 2.0) Organization Service Types and States .....	142
(CM 2.0) Organization Types .....	143
(CM 2.0) Organization Service Entity Diagram .....	145
(CM 2.0) Program Service .....	145
(CM 2.0) Program Service Description and Assumptions .....	147
(CM 2.0) Program Service Types and States .....	148
(CM 2.0) Program Types .....	149
(CM 2.0) Program Service Entity Diagram .....	149
(CM 2.0) Proposal Service .....	150
(CM 2.0) Proposal Service Description and Assumptions .....	152
(CM 2.0) Proposal Types and States .....	152
(CM 2.0) KEW Proposal Related Document Types .....	153
(CM 2.0) Proposal Types .....	154
(CM 2.0) Reference Types .....	154
(CM 2.0) Proposal Service Entity Diagram .....	155
(CM 2.0) Search Service .....	155
(CM 2.0) Search Service Description and Assumptions .....	157
(CM 2.0) Search Service Entity Diagram .....	157
(CM 2.0) Search Types and States .....	158
(CM 2.0) Formatted View of ATP Searches .....	160
(CM 2.0) Formatted View of CLU Searches .....	171
(CM 2.0) Formatted View of Comment Searches .....	251
(CM 2.0) Formatted View of EM Searches .....	252
(CM 2.0) Formatted View of LO Searches .....	254
(CM 2.0) Formatted View of LRC Searches .....	266
(CM 2.0) Formatted View of Organization Searches .....	269
(CM 2.0) Formatted View of Proposal Searches .....	292
(CM 2.0) Formatted View of Statement Searches .....	295
(CM 2.0) Formatted View of Subject Code Searches .....	297
(CM 2.0) State Service .....	299
(CM 2.0) State Service Description and Assumptions .....	300
(CM 2.0) State Service Implementation Notes .....	301
(CM 2.0) State Service FAQ .....	301
(CM 2.0) State service Types and States .....	302
(CM 2.0) State Types and States .....	302
(CM 2.0) State Change States .....	304
(CM 2.0) State Change Types .....	305
(CM 2.0) State Constraint States .....	305
(CM 2.0) State Constraint Types .....	305
(CM 2.0) State Propagation States .....	306
(CM 2.0) State Propagation Types .....	306
(CM 2.0) State Service Entity Diagram .....	306
(CM 2.0) Type Service .....	307
(CM 2.0) Type Service Description and Assumptions .....	308
(CM 2.0) TypeTypeRelation Types and States .....	309
(CM 2.0) TypeType Relations States .....	309
(CM 2.0) TypeType Relation Types .....	309
(CM 2.0) Type Service Entity Diagram .....	310
(CM 2.0) Version Management Service .....	311
(CM 2.0) Version Management Service Description and Assumptions .....	312
(CM 2.0) Version Management Service Entity Diagram .....	314

# CM Services 2.0 Documentation

The service contracts used in Kuali Student define the integration points between modules and the core infrastructure and between the modules themselves. Unlike systems where there are separate internal (or private) interfaces and external (or public) interfaces, the Kuali Student System uses services for both, allowing various modules or service implementations to be substituted. Although there is no part of the Kuali Student System that is required, the Service Contracts must be used in the prescribed ways. Therefore, the design and stability of the service contracts is critical.

There are two classifications of service contracts in Kuali Student:

- Business Services: Business services contain the capabilities required to accomplish processes within a specific business domain.
- Entity Services: Entity services are business agnostic in that contain capabilities that are required across multiple domains within the enterprise.

## How to Use This Index

This page provides an overview of the service contracts for Kuali Student including the most current version and links to the detailed service contract. In the table below, click on **Javadoc** to view the contract in Javadoc format, or click on **Description** to view the contract in a more human-readable form on the Wiki.

Service	Documentation	Type	Version	Description
Academic Time Period (ATP) Service	<a href="#">Description</a> or <a href="#">Javadoc</a>	Class Core	2.0	The Academic Time Period Service manages Academic Time Periods (ATPs) and their associated Milestones. The service provides a flexible but structured way to define the various time frames that are used throughout the definition, offering and scheduling of Learning Units. This is a catalog service with basic operations.
Canonical Learning Unit (CLU) Service	<a href="#">Description</a> or <a href="#">Javadoc</a>	Class I	2.0	The Canonical Learning Unit Service supports the management of Canonical Learning Units (CLUs). This includes the development and approval of new Learning Units and substantive changes to existing Learning Units. A Learning Unit (LU) is any learning-related activity that needs to be tracked by the institution or the learner. Examples of learning units include the traditional curriculum of courses and degrees, professional and extension programs, and non-academic activities such as leadership development and service learning. All credit and non-credit learning activities, whether traditional or non-traditional, are learning units.

Comment Service	Description or Javadoc	Class Core	2.0	The Comment Service allows for the creation and management of user comments and tags associated with other objects across the system. There is no expectation that the objects know anything about the tags or comments; therefore objects can be deleted from the system even though tags or comments referencing those object may exist.
Course Service	Description or Javadoc	Class II	2.0	A course is a specified area of knowledge contained and taught independently from other areas of knowledge that may or may not be related. Courses typically have specific learning objectives, defined student learning outcomes and assessments. They can be recognized for academic credit or not. Courses can be delivered in various instructional formats such as lecture, lab, discussion, seminar, colloquium, etc. A course would utilize at least one instructional format, but often will use more than one.
Dictionary Service 1.0	Description or Javadoc	Class I	1.0	The Dictionary Service subinterface provides information as to the data configuration of the service. This information is limited to things such as cardinality, field length, allowed values, etc. and does not include direct information as to cross-field validation. All services referring to dynamic attributes must extend this interface.
Dictionary Service 1.1	Description or Javadoc	Class I	1.1	Added support for marking the "other" field when a cross-field constraint failed and support for warnings.
Document Service	Description or Javadoc	Class Core	2.0	The Document Service supports the management of document objects. Relations between stored documents and external entities are managed through the respective entity service.

Enumeration Mgmt Service	Description or Javadoc	Class Core	2.0	Enumeration Management service is only accessed by authorized callers configuring some piece of the system. The code identifies the value within an enumeration.
Learning Objective Service	Description or Javadoc	Class I	2.0	The Learning Objective Service supports the management of formal learning objectives. This is primarily a catalog service with basic operations.
Learning Result Catalog Service	Description or Javadoc	Class I	2.0	The Learning Result Catalog service supports the management of learning results. It provides the basis for defining the result types and values that can be associated with the catalog LU (CLU), the offered LU (LUI), and the specific student (LPR).
Message Service	Description or Javadoc	Class Core	2.0	Manages locale specific messages for internationalization. A message is unique in the context of <i>locale</i> and <i>group</i> . The <i>locale</i> parameter allows for operations pertaining to messages whose locale is different from that of <i>ContextInfo.locale</i> information.
Organization Service	Description or Javadoc	Class Core	2.0	This service manages organizational units that have a relationship to the institution. The organizations may be internal and include officially recognized organizations (e.g. Departments, Faculties, Schools) or unofficial organizations (e.g. clubs or student groups), or they may be external organizations (e.g. companies, other institutions, government, associations). This service also manages the relationships between people and organizations.

Program Service	Description or Javadoc	Class II	2.0	A program is a prescribed grouping of learning units such as courses, activities, competencies, learning objectives, and/or institutional/core requirements that lead to a recognized body of knowledge. Programs may also include requirements as to the order, timing, performance, and results (e.g. GPA) of the grouping of learning units that make up a program. The end result of a completed program may be an acknowledged level of accomplishment (e.g. a major, minor or concentration) or a credential (e.g. certificate, degree, etc.)
Proposal Service	Description or Javadoc	Class I	2.0	The Proposal Service supports the management of Proposals. The proposal is general and can be associated with any arbitrary entity in the system, for example, a learning unit. Proposals have types that capture the activity being proposed, for example, Creating a new course or Updating the prefix for a set of courses or Launching a new student club.

Search Service	Description or Javadoc	Class I	2.0	<p>The Search Service supports the creation of typed searches that query the underlying data store and return results in tabular form. Conceptually it can be viewed as a wrapper for predefined (typed) SQL statements. It has two main purposes:</p> <ol style="list-style-type: none"> <li>1. The definitions of searches to meet specific purposes without modifying the contract.</li> <li>2. The definition of searches that return data other than objects and their attributes, for example counts or sums.</li> </ol>
State Service	Description or Javadoc	Class Core	2.0	State service is used for managing the following: valid States of an object, constraints associated with a state change and propagation triggered by a State change. It also describes a way to manage constraint and propagation of States.
Statement Service	Description or Javadoc	Class Core	1.0	The Statement Service subinterface supports the management of statements, requirement components, and their attachment to objects.
Type Service	Description or Javadoc	Class Core	2.0	Type service is Kuali Student's generic way of reading type information. Type service exposes information about a specific type and also the relationship between types.

Version Management Service	Description or Javadoc	Class Core	2.0	<p>The Version Management Service inspects versions of an object. The Version Management Service is used in conjunction with another service to examine the version history of an object. This service provides access to the identifiers to retrieve older or future versions of objects from their respective services.</p> <p>The Version Management Service is only useful when used in conjunction with another service that supports versioning.</p>
----------------------------	------------------------	------------	-----	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## (CM 2.0) Academic Time Period (ATP) Service

- Class
- Description
- Entity Diagram
- Service Contract Documentation
- Type and State Configuration
  - Calendar Types
  - Term Types Grouping

### Class

Academic Time Period is a Class I service that manages time periods and milestones.

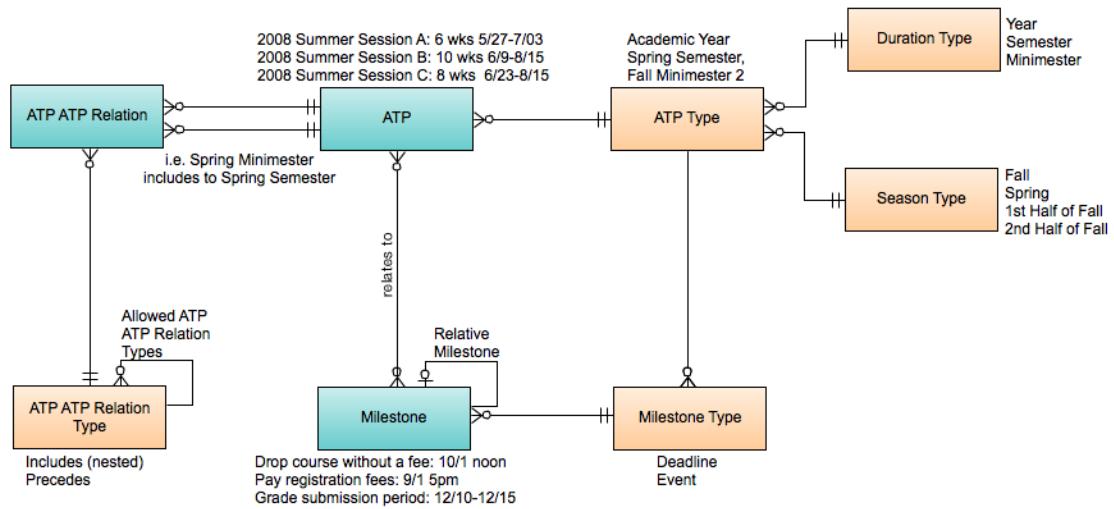
### Description

The Academic Time Period Service manages Academic Time Periods (ATPs) and their associated Milestones. The service provides a flexible but structured way to define the various time frames that are used throughout the definition, offering and scheduling of Learning Units. This is a catalog service with basic operations.

[More...](#)

### Entity Diagram

## Academic Time Period Service Entity Diagram



More...

## Service Contract Documentation

Go to Academic Time Period Service Contract Documentation

## Type and State Configuration

### Calendar Types

ATP Type Key	ATP Type Name	ATP Type Description	Typical Usage	Season Type	Duration Type	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions

kuali.atp.type.AcademicCalendar	Academic Calendar	Academic Calendar	Annual	AY	Year	Program Calendar	Configured	Y	Y	?	How do we support an academic calendar for an 18 month program?  ==> Perhaps an Academic Calendar Type can be associated with more than one Atp Type (similar to Credential Program can map to bachelors or masters or doctorate credential types)... if so we need a grouping .
kuali.atp.type.HolidayCalendar	Holiday Calendar	Holiday Calendar	Annual	AY	Year		Configured	Y	Y		

## Term Types Grouping

kuali.atp.type.group.term -- The list of ATP types that are considered terms in that courses may be offered during them.

ATP Type Key	Sub-Term of what term?	ATP Type Name	ATP Type Description	Typical Usage	Season Type	Duration Type	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions	Maps to Reference Institution	No tes re: Reference Institution
					kuali.atp.type.group.season	kuali.atp.type.group.duration							

kuali.at p.type. Fall		Fall	Fall Semester	Credit Course	Fall	Semester		Configured	Y	Y		Yes	Duration: Type: Quarter Season: Autumn Name: Autumn Quarter
kuali.at p.type. FallSpr ing		Fall-Spr ing	Fall & Spring Semesters	Credit Course	Fall	2 Semesters		Placeholder	N	N	Used for those year long course s that start in the fall and end in the spring		
kuali.at p.type. HalfFal l1	Fall	Half Fall 1	1st Half Semester in Fall	Credit Course	Fall 1	Half Semester		Configured	Y	Y			
kuali.at p.type. HalfFal l2	Fall	Half Fall 2	2nd Half Semester in Fall	Credit Course	Fall 2	Half Semester		Configured	Y	Y			
kuali.at p.type. Winter		Winter	Winter Activity Period	Credit Course	Winter	Period		Configured	Y	Y		Yes	Duration: Type: Quarter Season: Winter Name: Winter Quarter
kuali.at p.type. Spring		Spring	Spring Semester	Credit Course	Spring	Semester		Configured	Y	Y		Yes	Duration: Type: Quarter Season: Spring Name: Spring Quarter

kuali.at.p.type.HalfSpring1	Spring	Half Spring 1	1st Half Semester in Spring	Credit Course	Spring 1	Half Semester		Configured	Y	Y			
kuali.at.p.type.HalfSpring2	Spring	Half Spring 2	2nd Half Semester in Spring	Credit Course	Spring 2	Half Semester		Configured	Y	Y			
kuali.at.p.type.Spring Break	Spring	Spring Break	Spring Break Experiential Term	Credit Course	Spring Break	Week		Configured	Y	Y	Do not confuse this with the Milestone that defines the Spring Break period where typically no classes are held. This is actually an example of a term that can be used for special week long experiential or service learning programs that over spring break.		
kuali.at.p.type.Summer		Summer	Summer	Credit Course	Summer	Term		Configured	Y	Y	Used to group periods for display and reporting	Yes	Duration Type: Quarter Season: Summer Name: Summer Quarter

kuali.at p.type. Summ erEve	Summ er	Summ er Eve	Summ er Evenin g	Credit Cours e	Summ er	Term		Config ured	Y	Y	Used to group peri ods for display and reporti ng and for LUI creatio n only for evenin g classe s		
kuali.at p.type. Sessio n1	Summ er	Sessio n 1	1st Summ er Sessio n	Credit Cours e	Summ er 1	Sessio n		Config ured	Y	Y		Yes	Duration Type: Subter m Seaso n: Summ er Name: Summ er, A-Ter m Notes: Runs concur rent with full Summ er Quarte r, NO overla p with B-Ter m
kuali.at p.type. Mini-m ester1 A	Sessio n 1	Mini-m ester 1A	Summ er Mini-m ester 1A	Credit Cours e	Summ er 1A	Mini-m ester		Config ured	Y	Y			
kuali.at p.type. Mini-m ester1 B	Sessio n 1	Mini-m ester 1B	Summ er Mini-m ester 1B	Credit Cours e	Summ er 1B	Mini-m ester		Config ured	Y	Y			

kuali.at p.type. Session2	Summ er	Sessio n 2	2nd Summ er Sessio n	Credit Cours e	Summ er 2	Sessio n		Config ured	Y	Y		Yes	Durati on Type: Subter m Seaso n: Summ er Name: Summ er, B-Ter m Notes: Runs concur rent with full Summ er Quarte r, NO overla p with A-Ter m
kuali.at p.type. Mini-m ester2C	Sessio n 2	Mini-m ester 2C	Summ er Mini-m ester 2C	Credit Cours e	Summ er 2C	Mini-m ester		Config ured	Y	Y			
kuali.at p.type. Mini-m ester2D	Sessio n 2	Mini-m ester 2D	Summ er Mini-m ester 2D	Credit Cours e	Summ er 2D	Mini-m ester		Config ured	Y	Y			
kuali.at p.type. SessionG1	Summ er	Sessio n G1	1st Grad Summ er Sessio n	Credit Cours e	Summ er 1	Sessio n		Config ured	Y	Y			
kuali.at p.type. SessionG2	Summ er	Sessio n G2	2nd Grad Summ er Sessio n	Credit Cours e	Summ er 2	Sessio n		Config ured	Y	Y			
kuali.at p.type. Adhoc		Adhoc	Ad hoc sessio n	Lui	Any	TBD	On-off	Placeh older	N	N		Used for one-off course s that are taught on their own sched ule	

More...

# (CM 2.0) Academic Time Period Service Description and Assumptions

## Academic Time Period Service

### Description

The Academic Time Period Service manages Academic Time Periods (ATPs) and their associated Milestones. The service provides a flexible but structured way to define the various time frames that are used throughout the definition, offering and scheduling of Learning Units. This is a catalog service with basic operations.

Milestones are a single date/time or a range between two date/times. A Milestone can represent an event such as a deadline for application submission (single date) or a period such as Grade Submission (date range). Milestones are managed independently from an ATP and may be mapped to one or more ATPs. There is not an explicit service entity for the ATP - Milestone relation.

An Academic Time Period (ATP) has a start date/time and an end date/time. These values do not constrain the related Milestones. In other words, an ATP for Fall semester may have a pre-registration Milestone with a date in May.

ATP relationships such as nesting is represented by ATP ATP Relationships. An ATP may be defined for "Spring" and another ATP for "Spring minimester 2."

### Key Concepts

- Milestones are independently managed and mapped to ATPs. The relationship is implied directly through service operations and explicit relationship entities.
- A Milestone with a day but no time component has an attribute of "all day."
- A Milestone with a "range" attribute requires an end date, null not allowed.
- ATP relationships capture the concept of nested Terms and mapping of Terms to Academic Calendars.

### Assumptions

- Type setup is part of configuration and not managed specifically through the service, including atpType (using durationType and seasonType), atpAtpRelationType and MilestoneType.
- Date comparisons are always inclusive, after means  $\geq$  date and before means  $\leq$  date, **unless explicitly stated otherwise**.

### Deferred Design

- Logical Milestones
- Revisit the Milestone Types in light of Process/Check
  - We may also need an "any" or "infinite" ATP Type

### Implementation Notes

(CM 2.0) Storing and Querying Milestone Dates

## (CM 2.0) Storing and Querying Milestone Dates

### General Rules

1. If the IsAllDay is true then start is set to start of day (00:00:00.0) and end date is set to end of day (12:59.59.9)
2. If the is Date range is false then end date is always null

### Matrix

is All Day	Is Date Range	Start Date	End Date	Example Query
False	False	Not truncated on storage	set to null on storage	startDate = searchDate1 startDate > searchDate1 startDate between searchDate1 and searchDate2
True	False	Set to start of day on storage	set to null on storage	startDate = searchDate1 startDate > searchDate1 startDate between searchDate1 and searchDate2)

False	True	Not changed on storage	Not changed on storage	searchDate1 between startDate and endDate searchDate1 > startDate  (overlap) searchDate1 between startDate and endDate or startDate between searchDate1 and searchDate2
True	True	Set to start of day on storage	Set to End of Day on storage	searchDate1 between startDate and endDate searchDate1 > startDate  (overlap) searchDate1 between startDate and endDate or startDate between searchDate1 and searchDate2

### Overlap logic

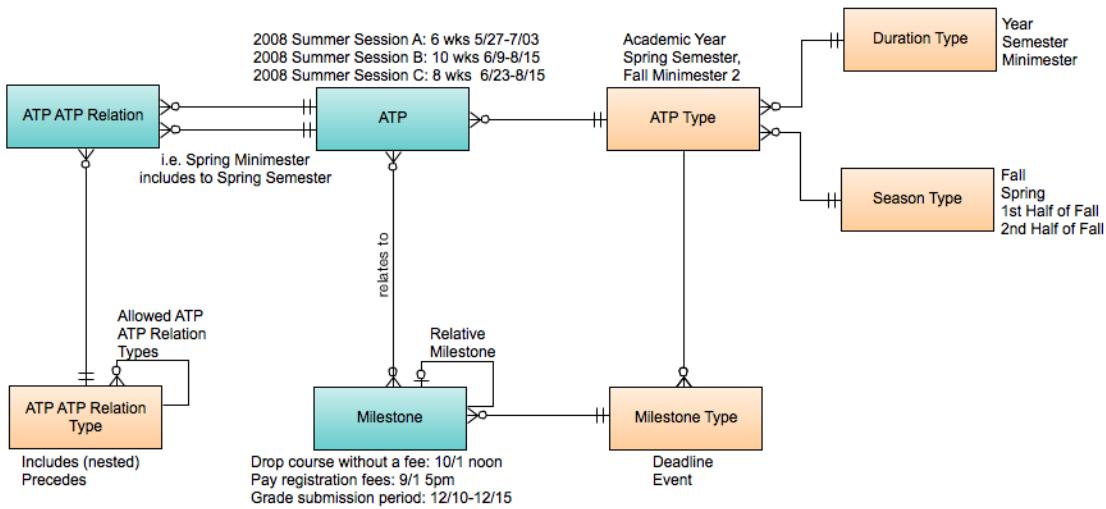
			paramStart			paramEnd			Comment	Want to select	Selected by start between :paramStart and :paramEnd	Selected by :paramStart between start and end
Range	start	end							before			
Range			start	end					inside	Yes	x	
Range					start	end			after			
Range			start			end			starts in middle	Yes	x	
Range		start		end					ends in middle	Yes		x
Range		start				end			larger than range	Yes		x
point		start							before			
point			start						during	Yes	x	
point					start				after			

### (CM 2.0) Academic Time Period Service Entity Diagram

- Academic Time Period Service Entity Diagram
- Database ERD

#### Academic Time Period Service Entity Diagram

## Academic Time Period Service Entity Diagram



## Database ERD

STUDENT:Database ER Diagram

KSEN\_ATP\_DDL\_TABLES.ddl

KSEN\_ATP\_DDL\_CONSTRAINTS.ddl

## (CM 2.0) Academic Time Period Types and States

- ATP Types
  - Calendar Types
  - Term Types Grouping
  - Program Related Types
  - When Course Offered Types
  - Reporting Types
  - Dynamic Attribute Types
- ATP States
  - ATP States
  - Milestone States
- Milestone Types
  - Key Date Milestones
  - Seatpool Milestones
  - Curriculum Milestones
  - Holidays
    - Non-Instructional Holidays
    - Instructional Holidays
  - Academic Calendar Event Milestones
  - Graduation Milestones
  - Admissions Milestones
  - Orientation Milestones
  - Billing Milestones
  - Financial Aid Milestones
  - Scheduling Milestones
  - Tuition Calculation Milestones
  - Misc
- Duration Types
- Seasonal Types
- Relationships
  - ATP to ATP Types
  - ATP to ATP States

Allowed ATP to ATP Types

## References

- KU Calendar Normalized.xls Original ATPs and milestones based on the Old Kuali Reference University Work
- Academic Calendar - Milestones – new ATEAM feedback on this page

## ATP Types

### Calendar Types

ATP Type Key	ATP Type Name	ATP Type Description	Typical Usage	Season Type	Duration Type	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.type.AcademicCalendar	Academic Calendar	Academic Calendar	Annual	AY	Year	Program Calendar	Configured	Y	Y	<p>💡 How do we support an academic calendar for an 18 month program?</p> <p>==&gt; Perhaps an Academic Calendar Type can be associated with more than one Atp Type (similar to Credential Program can map to bachelors or masters or doctorate credential types)... if so we need a grouping .</p>
kuali.atp.type.HolidayCalendar	Holiday Calendar	Holiday Calendar	Annual	AY	Year		Configured	Y	Y	

### Term Types Grouping

kuali.atp.type.group.term -- The list of ATP types that are considered terms in that courses may be offered during them.

ATP Type Key	Sub-Term of what term?	ATP Type Name	ATP Type Description	Typical Usage	Season Type	Duration Type	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions	⭐ Maps to Reference Institution	⭐ Notes re: Reference Institution
kuali.atp.type.Fall		Fall	Fall Semester	Credit Course	Fall	Semester		Configured	Y	Y		Yes	Duration Type: Quarter Season: Autumn Name: Autumn Quarter
kuali.atp.type.FallSpring		Fall-Spring	Fall & Spring Semesters	Credit Course	Fall	2 Semesters		Placeholder	N	N	Used for those year long courses that start in the fall and end in the spring		
kuali.atp.type.HalfFall1	Fall	Half Fall 1	1st Half Semester in Fall	Credit Course	Fall 1	Half Semester		Configured	Y	Y			
kuali.atp.type.HalfFall2	Fall	Half Fall 2	2nd Half Semester in Fall	Credit Course	Fall 2	Half Semester		Configured	Y	Y			
kuali.atp.type.Winter		Winter	Winter Activity Period	Credit Course	Winter	Period		Configured	Y	Y		Yes	Duration Type: Quarter Season: Winter Name: Winter Quarter

kuali.at p.type. Spring		Spring	Spring Semester	Credit Cours e	Spring	Semester		Config ured	Y	Y		Yes	Duration Type: Quarter Season: Spring Name: Spring Quarte r
kuali.at p.type. HalfSp ring1	Spring	Half Spring 1	1st Half Semes ter in Spring	Credit Cours e	Spring 1	Half Semes ter		Config ured	Y	Y			
kuali.at p.type. HalfSp ring2	Spring	Half Spring 2	2nd Half Semes ter in Spring	Credit Cours e	Spring 2	Half Semes ter		Config ured	Y	Y			
kuali.at p.type. Spring Break	Spring	Spring Break	Spring Break Experi ential Term	Credit Cours e	Spring Break	Week		Config ured	Y	Y	Do not confus e this with the Milesto ne that define s the Spring Break period where typicall y no classe s are held. This is actuall y an examp le of a term that can be used for special week long experi ential or service learnin g progra ms that over spring break.		

kuali.at p.type. Summ er		Summ er	Summ er	Credit Cours e	Summ er	Term		Config ured	Y	Y	Used to group peri ods for display and reporti ng	Yes	Durati on Type: Quarte r Seaso n: Summ er Name: Summ er Quarte r
kuali.at p.type. Summ erEve	Summ er	Summ er Eve	Summ er Evenin g	Credit Cours e	Summ er	Term		Config ured	Y	Y	Used to group peri ods for display and reporti ng and for LUI creatio n only for evenin g classe s		
kuali.at p.type. Sessio n1	Summ er	Sessio n 1	1st Summ er Sessio n	Credit Cours e	Summ er 1	Sessio n		Config ured	Y	Y		Yes	Durati on Type: Subter m Seaso n: Summ er Name: Summ er, A-Ter m Notes: Runs concur rent with full Summ er Quarte r, NO overla p with B-Ter m
kuali.at p.type. Mini-m ester1 A	Sessio n 1	Mini-m ester 1A	Summ er Mini-m ester 1A	Credit Cours e	Summ er 1A	Mini-m ester		Config ured	Y	Y			

kuali.at p.type. Mini-m ester1 B	Sessio n 1	Mini-m ester 1B	Summ er Mini-m ester 1B	Credit Cours e	Summ er 1B	Mini-m ester		Config ured	Y	Y			
kuali.at p.type. Sessio n2	Summ er	Sessio n 2	2nd Summ er Sessio n	Credit Cours e	Summ er 2	Sessio n		Config ured	Y	Y		Yes	Duration Type: Subter m Seaso n: Summ er Name: Summ er, B-Ter m Notes: Runs concur rent with full Summ er Quarte r, NO overla p with A-Ter m
kuali.at p.type. Mini-m ester2 C	Sessio n 2	Mini-m ester 2C	Summ er Mini-m ester 2C	Credit Cours e	Summ er 2C	Mini-m ester		Config ured	Y	Y			
kuali.at p.type. Mini-m ester2 D	Sessio n 2	Mini-m ester 2D	Summ er Mini-m ester 2D	Credit Cours e	Summ er 2D	Mini-m ester		Config ured	Y	Y			
kuali.at p.type. Sessio nG1	Summ er	Sessio n G1	1st Grad Summ er Sessio n	Credit Cours e	Summ er 1	Sessio n		Config ured	Y	Y			
kuali.at p.type. Sessio nG2	Summ er	Sessio n G2	2nd Grad Summ er Sessio n	Credit Cours e	Summ er 2	Sessio n		Config ured	Y	Y			

kuali.atp.type.Adhoc		Adhoc	Ad hoc session	Lui	Any	TBD	On-off	Placeholder	N	N	Used for one-off courses that are taught on their own schedule		
----------------------	--	-------	----------------	-----	-----	-----	--------	-------------	---	---	----------------------------------------------------------------	--	--

## Program Related Types

ATP Type Key	ATP Type Name	ATP Type Description	Typical Usage	Season Type	Duration Type	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.type.UndergradProgram	Undergrad Program	Four Year Undergraduate	Program	Four Year Cycle	Four Years		Configured	Y	Y	
kuali.atp.type.FreshmanYear	Freshman Year	Freshman Year	Program	Year 1	Year		Configured	Y	Y	
kuali.atp.type.FreshmanYearTerm1	Freshman Year Term 1	1st Term Freshman	Program	Term 1	Semester		Configured	Y	Y	
kuali.atp.type.FreshmanYearTerm2	Freshman Year Term 2	2nd Term Freshman	Program	Term 2	Semester		Configured	Y	Y	
kuali.atp.type.SophomoreYear	Sophomore Year	Sophomore Year	Program	Year 2	Year		Configured	Y	Y	
kuali.atp.type.SophomoreYearTerm1	Sophomore Year Term 1	1st Term Sophomore	Program	Term 1	Semester		Configured	Y	Y	
kuali.atp.type.SophomoreYearTerm2	Sophomore Year Term 2	2nd Term Sophomore	Program	Term 2	Semester		Configured	Y	Y	
kuali.atp.type.JuniorYear	Junior Year	Junior Year	Program	Year 1	Year		Configured	Y	Y	
kuali.atp.type.JuniorYearTerm1	Junior Year Term 1	1st Term Junior	Program	Term 1	Semester		Configured	Y	Y	
kuali.atp.type.JuniorYearTerm2	Junior Year Term 2	2nd Term Junior	Program	Term 2	Semester		Configured	Y	Y	
kuali.atp.type.SeniorYear	Senior Year	Senior Year	Program	Year 4	Year		Configured	Y	Y	

kuali.atp.type.SeniorYearTerm1	Senior Year Term 1	1st Term Senior	Program	Term 1	Semester		Configured	Y	Y	
kuali.atp.type.SeniorYearTerm2	Senior Year Term 2	2nd Term Senior	Program	Term 2	Semester		Configured	Y	Y	

#### When Course Offered Types

ATP Type Key	ATP Type Name	ATP Type Description	Typical Usage	Season Type	Duration Type	Aliases	Agreed Upon?	In Constant File?	In Database ?	Comments/Questions
kuali.atp.type.EvenYears	Even Years	Even Numbered Years	Offering	Even Years	Year		Configured	Y	Y	
kuali.atp.type.FallEvenYears	Fall Even Years	Fall Even Years	Credit Course	Fall	Semester		Configured	Y	Y	
kuali.atp.type.SpringEvenYears	Spring Even Years	Spring Even Years	Credit Course	Spring	Semester		Configured	Y	Y	
kuali.atp.type.OddYears	Odd Years	Odd Numbered Years	Offering	Odd Years	Year		Configured	Y	Y	
kuali.atp.type.FallOddYears	Fall Odd Years	Fall Odd Years	Credit Course	Fall	Semester		Configured	Y	Y	
kuali.atp.type.SpringOddYears	Spring Odd Years	Spring Odd Years	Credit Course	Spring	Semester		Configured	Y	Y	

#### Reporting Types

ATP Type Key	ATP Type Name	ATP Type Description	Typical Usage	Season Type	Duration Type	Aliases	Agreed Upon?	In Constant File?	In Database ?	Comments/Questions
kuali.atp.type.AY	AY	Full Academic Year	Annual	AY	Year		Configured	Y	Y	At present only used to group terms for display and reporting, not for LUI creation

kuali.atp.type.FY	FY	Fiscal Year	Annual	FY	Year		Configured	Y	Y	At present only used to group terms for display and reporting, not for LUI creation
-------------------	----	-------------	--------	----	------	--	------------	---	---	-------------------------------------------------------------------------------------

### Dynamic Attribute Types

Dynamic Attribute Type Key	Type Name	Type Description	Typical Usage	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.attribute.campus.key	Campus Key	Holds keys that indicate associated campuses	On Holiday Calendars	Location	Configured - Core Slice	N	N	

### ATP States

#### ATP States

Process Key	
kuali.atp.lifecycle	

State Key	State Name	State Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.state.Draft (initial)	Draft	Indicates that this Time Period is just tentative	Planned, Proposed, Tentative	Configured	Y	Y	 do we need/want a "rolled over" state or is draft good enough?
kuali.atp.state.Official	Official	Indicates that this Time Period has been established	Actual, Real, Publishable	Configured	Y	Y	There is no state to indicate that this is the current one. That is determined by the dates. This is still the Official academic calendar or term for the period to which it applies

### Milestone States

## Process Key

kuali.milestone.lifecycle

State Key	State Name	State Description	Aliases	Status	In Constant File?	In Database?	Comments/Questions
kuali.milestone.state.Draft (initial)	Draft	Indicates that this Milestone is just tentative	Planned, Proposed, Tentative	Configured	Y	Y	
kuali.milestone.state.Official	Official	Indicates that this Milestone has been established	Actual, Real	Configured	Y	Y	

## Milestone Types

### Key Date Milestones

kuali.milestone.type.group.keydate-- The list of milestone types that are used for Key Dates that are associated with a term.

Milestone Type Key	Hardened @ Class II Level ?	Is a Range?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Calculation Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions	★ Maps to Reference Institution	★ Notes re: Reference Institution

kuali.atp.milestone.AdvancedRegistrationPeriod		TRUE	1	Advance Registration Period	Advance Registration Period	Preeregistration period	Registrar		First two weeks in November for Spring, Last week in March to 1st week in April for Fall and Summer. Fall and Summer are the same time period	Configured - Core Slice	Y	N		Y	Used as a Date Range UW has TWO "Advance" Periods - Periods 1 and 2	
kuali.atp.milestone.RegistrationPeriod	TRUE	TRUE	2	Registration Period	Registration Period	Course Selection Period	Registrar		Typically the first week of the semester	Configured - Core Slice	Y	N	?	Is there difference between the end of the registration period and the Course Selection Period End Date below?	Y	UW Period 3

kuali.atp.milestone.Registration.Period1			Registration Period 1	Freshmen		Registrar		Configured	Y	Y	<b>KSENROLL-2</b> <b>806</b> - Configure "Approved" Key Dates in Constants File (Closed)	<b>KSENROLL-2</b> <b>807</b> - Configure "Approved" Key Dates in Database (Closed)	Implementing Institution Needs
kuali.atp.milestone.Registration.Period2			Registration Period 2	Sophomore		Registrar		Configured	Y	Y	<b>KSENROLL-2</b> <b>806</b> - Configure "Approved" Key Dates in Constants File (Closed)	<b>KSENROLL-2</b> <b>807</b> - Configure "Approved" Key Dates in Database (Closed)	Implementing Institution Needs
kuali.atp.milestone.Registration.Period3			Registration Period 3	Juniors		Registrar		Configured	Y	Y	<b>KSENROLL-2</b> <b>806</b> - Configure "Approved" Key Dates in Constants File (Closed)	<b>KSENROLL-2</b> <b>807</b> - Configure "Approved" Key Dates in Database (Closed)	Implementing Institution Needs

kuali.atp.milestone.Registration.Period4				Registration Period 4	Seniors		Registrar		Configured	Y	Y	<input checked="" type="checkbox"/> <b>KSENROLL-2</b> <b>806</b> - Configure "Approved" Key Dates in Constants File (Closed)	
												<input checked="" type="checkbox"/> <b>KSENROLL-2</b> <b>807</b> - Configure "Approved" Key Dates in Database (Closed)	
				Late Registration Period 1					Configured	Y	Y	Implementing Institution Needs	
kuali.atp.milestone.Registration.Period5				Late Registration Period 1					Configured	Y	Y	Dan E. found these in the Data base	
kuali.atp.milestone.Registration.Period6				Late Registration Period 2					Configured	Y	Y	Dan E. found these in the Data base	
kuali.atp.milestone.Registration.Period7				Schedule Changes					Configured	Y	Y	Dan E. found these in the Data base	
kuali.atp.milestone.Registration.BeginsforMBA		2.1	Registration Begins for MBA	MBA Registration begins		Registrar	Open	Fall only the last week in August	Configured	Y	Y	Is this really "registration period" but on the MBA calendar?	
kuali.atp.milestone.Registration.BeginsNonDegreree		2.2	Registration Begins Non Degreree	Registration for non-degree seeking students		Registrar	Open	Fall only 1st of September	Configured	Y	Y		

kuali.atp.milestone.Registration BeginsTransfer			2.3	Registration Begins Transfer	Registration for Transfer Students		Registar	Open	Typically the day after move-in date	Configured	Y	Y			
kuali.atp.milestone.Instructional Period	TRUE	TRUE	3	Instructional Period	Dates between which classes should meet	Classes begin , and end dates First meeting date and last meeting date of term	Registar	Open	For Fall the day after labor day, for Spring the day after MLK day and the last day For fall the Friday the week before Christmas, for Spring, the Friday the week before Memorial Day	Configured	Y	Y		Yes	
kuali.atp.milestone.Course SelectionPeriod End	TRUE		4	Course Selection Period	Course Selection Period ends	Last date to Add Date	Registar	Deadline	2 weeks into the term	Configured	Y	Y	Is this what I've called Add Date ? Is this the same as the end date of the "registration period"?	Yes	

kuali.atp.milestone.DropDeadlineWithoutRecord			5	Drop Deadline Without Record	Deadline to drop w/o "W"	Late Drop Date	Registar	Deadline	Only applies to Summer and Winter terms, set approximately halfway through the term.	Configured	Y	Y		Yes	
kuali.atp.milestone.DropDate	TRUE		6	Drop Date	Drop Period ends	Last date to withdraw from a class	Registar	Deadline	The Friday 6 weeks from the start of the Semester, only applies to Semesters not Summer and Winter	Configured	Y	Y			
kuali.atp.milestone.PostGradesMidterm			7	Post Grades Midterm	Mid-term Grades Posted		Registar	Process Run	15th of October for fall.	Configured	Y	Y	KU does not specify midterm grades for spring. Was this intentional?		
kuali.atp.milestone.MailProgressReports			8	Mail Progress Reports	Satisfactor y Progress Reports/Letters		Registar	Process Run	The same day the midterms grades are due	Configured	Y	Y	KU actually did not specify this date		

kuali.atp.milestone.Readin gPeriod		TRUE	10	Reading Period	Reading Days	Final Exam Preparation, Study Period	Registrar		Fall starting on the Wednesday two weeks before Christmas, for Spring the first full week in May	Configured	Y	Y			
kuali.atp.milestone.FinalExamPeriod	TRUE	TRUE	11	Final Exam Period	Final Examinations		Registrar		Based on Football Schedule	Configured	Y	Y		Yes	Currently only the Last Day of the Period is recorded
kuali.atp.milestone.GradesDue	TRUE		12	Grades Due	Grades Due		Registrar	Deadline	July 7 for Summer session	Configured	Y	Y	?	is there a grading Period? i.e a start date?	
kuali.atp.milestone.PostGrades			13	Post Grades	Grades Poste d	Publi sh Grades	Registrar	Process Run	Fall the day after Christmas, Spring the last Friday before Commencement, for the winter term the 10th of february	Configured	Y	Y			

## Seatpool Milestones

kuali.milestone.type.group.seatpool-- The list of milestone types that are used to drive the seatpool management process

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How Used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.at.p.milestone.firstdayofclasses			First Day of Classes						Configured	Y	Y	
kuali.at.p.milestone.lastdayofferegistration			Last Day of Registration						Configured	Y	Y	
kuali.at.p.milestone.endoffirstweekofclasses			End of First Week of Classes						Configured	Y	Y	
kuali.at.p.milestone.monthpriorstartofclasses			Month Prior to Start of Classes						Configured	Y	Y	

## Curriculum Milestones

kuali.milestone.type.group.curriculum-- The list of milestone types that are used to drive the curriculum management process

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.at.p.milestone.CoordinatorsKickoffMeeting		1	Coordinators Kickoff Meeting	Coordinators Kickoff Meeting		Registrar	Meeting	1st of October	Configured	Y	Y	

kuali.atp.miles tone.CurriculumCommitteeMeeting		2	Curriculum Committee Meeting	Curriculum Committee Meeting	after the deadline for proposals. Thereafter weekly every Tuesday, skipping for Christmas break, until the week after the final review period in April.	Registrar	Meeting	First meeting on the Tuesday the week	Configured	Y	Y
kuali.atp.miles tone.ProposalPeriod	TRUE	3	Proposal Period	Proposal Period		Registrar		Middle of September to the end of the next August	Configured - Core Slice	Y	N
kuali.atp.miles tone.MajorChangesDeadline		4	Major Changes Deadline	Deadline for Proposals with Major Changes		Registrar	Deadline	The same day as Final Exams for the fall term	Configured	Y	Y
kuali.atp.miles tone.MinorChangesDeadline		5	Minor Changes Deadline	Deadline for Proposals with Minor Changes		Registrar	Deadline	Around the 10th of January	Configured	Y	Y
kuali.atp.miles tone.LastMinuteProposalsDeadline		6	Last Minute Proposals Deadline	Deadline for Last Minute Proposals		Registrar	Deadline	Two weeks before changes need to be published on-line for that term	Configured	Y	Y

kuali.at p.miles tone.R evew Period	TRUE	7	Revie w Period	Period within which propos al are groupe d and review ed togeth er for approv al		Registr ar		3 of these in a year, one from Oct to the middle of februar y, 2nd one from the middle of februar y to the the 3rd week in March, and the final one from that period until the start of the next fall term	Config ured - Core Slice	Y	N	Used as Date Range
kuali.at p.miles tone.P ublish Chang esOnLi ne		8	Publis h Chang es On Line	Approv ed Cours e Chang es Publis hed On-Lin e		Registr ar	Proces s	The day before Advan ce registr ation opens	Config ured	Y	Y	Did not exist in KU but need somet hing like them

## Holidays

```
kuali.milestone.type.group.holidays -- The list of milestone types that are allowed  
for holidays
```

 The list should include both the instructional holidays and the non-instructional holidays below.

### ***Non-Instructional Holidays***

```
kuali.milestone.type.group.holidays.non-instructional -- The list of milestone types  
that are typically used for non-instructional holidays
```

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.at p.milestone.LaborDay		1	Labor Day	Labor Day		University	Date	1st Monday in September	Configured	Y	Y	
kuali.at p.milestone.VeteransDay		3	Veterans Day	Veterans Day		University	Date	November 11th.	Configured - Core Slice	N	Y	Added 1/10 TL
kuali.at p.milestone.VeteransDayObserved		3	Veterans Day Observed	Veterans Day Observed		University	Date	November 11th. If holiday falls on Saturday, observed on Friday. If holiday falls on Sunday, observed on Monday.	Configured - Core Slice	N	Y	Added 1/10 TL
kuali.at p.milestone.ThanksgivingBreak	TRUE	4	Thanks giving Break	Thanks giving Break		University	Open	4th Thursday-Sunday of November	Configured	Y	Y	Changed from Wed-Mon. 1/10 TL
kuali.at p.milestone.Christmas		5	Christmas Day	Christmas Day		University	Date	December 25th.	Configured - Core Slice	N	Y	Added 1/10 TL
kuali.at p.milestone.Christmas Observed		5	Christmas Day Observed	Christmas Day Observed		University	Date	December 25th. If holiday falls on Saturday, observed on Friday. If holiday falls on Sunday, observed on Monday.	Configured - Core Slice	N	Y	Added 1/10 TL

kuali.at p.milestone.Ne wYears Day		6	New Years Day	New Years Day		Univers ity	Date	January 1.	Configu red - Core Slice	N	Y	<span style="color:red">Added 1/10 TL</span>
kuali.at p.milestone.Ne wYears DayOb served		6	New Years Day Observed	New Years Day Observed		Univers ity	Date	January 1. If holiday falls on Saturday, observed on Friday. If holiday falls on Sunday , observed on Monday.	Configu red - Core Slice	N	Y	<span style="color:red">Added 1/10 TL</span>
kuali.at p.milestone.MLKDay		7	MLK Day	Martin Luther King, Jr. Day		Univers ity	Date	3rd Monday in January	Placeholder	N	N	<span style="color:red">Was MLKDayOb served. 1/10 TL</span>
kuali.at p.milestone.MLKDayO bserved			MLK Day Observed	Martin Luther King Jr. Day Observed		Univers ity	Date	3rd Monday in January	Configu red	Y	Y	
kuali.at p.milestone.President s Day		8	Preside nts Day	Preside nts Day		Univers ity	Date	3rd Monday in February	Configu red - Core Slice	N	Y	<span style="color:red">Added 1/10 TL</span>
kuali.at p.milestone.MemorialDay		10	Memori al Day	Memori al Day		Univers ity	Date	Last Monday in May	Configu red - Core Slice	N	Y	<span style="color:red">Was MemorialDayO bserve d. 1/10 TL</span>
kuali.at p.milestone.MemorialDayObse rved			Memori al Day Observed	Memori al Day Observed		Univers ity	Date	Last Monday in May	Configu red	Y	Y	
kuali.at p.milestone.Independenc eDay		11	Indepe ndence Day	Indepe ndence Day		Univers ity	Date	July 4th.	Configu red - Core Slice	N	Y	

kuali.at p.milestone.Indepe nienceDay Observed		11	Independence Day Observed	Independence Day Observed		University	Date	July 4th. If holiday falls on Saturday, observed on Friday. If holiday falls on Sunday, observed on Monday.	Configured	Y	Y	
kuali.at p.milestone.Ot herNon Instructi onalHol iday		12	Other Non-Instruc tional Holiday Day	Other Holiday Manag ed as Non-Instruc tional		University	Date		Placeholder	N	N	
kuali.at p.milestone.Ot herNon Instructi onalDa y		1	Other non-Instruc tional Day	A day on which instruct does not occur but is not typically consid ered a holiday					Placeholder	N	N	

### ***Instructional Holidays***

kuali.milestone.type.group.holidays.instructional -- The list of milestone types that are typically used for instructional holidays

Milesto ne Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Respo nsible Organi zation	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Q uestions
kuali.at p.milest one.Yo mKippur	??		Yom Kippur	Yom Kippur			Date		Example	N	N	UW: Occasion ally Non-Instruc tional
kuali.at p.milest one.Go odFrida y			Good Friday	Good Friday			Date	Friday before Easter	Example	N	N	

kuali.at p.milest one.Fal lBreak	TRUE		Fall Break	Fall Break		Univers ity	Open	Columb us Day Weeke nd	Configu red - Core Slice	Y	N	Used as a Date Range N/A for UW: Moved from non-ins tr. 1/10 TL
kuali.at p.milest one.Spr ingBrea k	True		Spring Break	Spring Break Begins		Univers ity	Open	Friday of the first full week in March and classes resume Monday followin g the week break.	Configu red - Core Slice	Y	N	Used as a Date Range N/A for UW: Moved from non-ins tr. 1/10 TL
kuali.at p.milest one.Ot herInstr uctional Holiday			Other Instructi onal Holiday	Other Holiday Manag ed as Instructi onal					Placeh older	N	N	Added 1/10 TL
kuali.at p.milest one.Col umbus Day			Columb us Day	Columb us Day		Univers ity	Date		Configu red	Y	Y	<b>803</b> - Co "Approve Instruction in Const Clo  <b>805</b> - Co "Approve Instruction in Database Clo  UMD Ref Data
kuali.at p.milest one.ele ctionda y			Election Day	Election Day		Univers ity	Date		Configu red	Y	Y	<b>803</b> - Co "Approve Instruction in Const Clo  <b>805</b> - Co "Approve Instruction in Database Clo  UMD Ref Data

kuali.at.p.milestone.rose.hashahnah			Rosh Hashanah	Rosh Hashanah		University	Date		Configured	Y	Y	803 - Co "Approve Instruction in Constant Close
												805 - Co "Approve Instruction in Database Close UMD Ref Data

### Academic Calendar Event Milestones

kuali.milestone.type.group.acal.event -- The list of milestone types that are used for academic calendar events

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.at.p.milestone.FamilyWeekend	TRUE	1	Family Weekend	Family Weekend	Parent Weekend	University		2nd Friday in october until that Sunday	Configured - Core Slice	Y	N	
kuali.at.p.milestone.Homecoming		2	Homecoming	Homecoming		University	Date	In October based on Football Schedule	Configured	Y	Y	
			Commencement	Graduation					Placeholder	N	N	
			Alumni Day						Placeholder	N	N	

This list should also include the types that are also in the Graduation Milestone group.

### Graduation Milestones

kuali.milestone.type.group.acal.event.graduation -- The list of milestone types that are typically used for non-instructional holidays

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.at p.milestone.GraduationApplicationDeadline		1	Graduation Application Deadline	Deadline to apply for Graduation		University	Deadline	Nov 15th for the fall, Summer August 1st	Configured	Y	Y	KU has no deadline for Spring which is unusual I think it is missing
kuali.at p.milestone.AlumniDay		1	Alumni Day	Alumni Day	Welcome back day	University	Date	2nd Saturday in May	Configured	Y	Y	Also an Event?
kuali.at p.milestone.Baccalaureate		2	Baccalaureate	Baccalaureate		University	Date	The day after alumni day	Configured	Y	Y	
kuali.at p.milestone.Commencement		3	Commencement	Commencement		University	Date	They day after Baccalaureate	Configured	Y	Y	
kuali.at p.milestone.LeaveofAbsenceBegin		4	Leave of Absence Begin	Leave of Absence Begin			Date		Configured	Y	Y	

#### Admissions Milestones

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.at p.milestone.AdmissionsApplicationsApplicationDueEarlyCycle		1	Admissions Application Due Early Cycle	Early Admissions Application Due Date		Admissions	Deadline	December 1st	Configured	Y	Y	
Obviously there are many many more...												

#### Orientation Milestones

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions

kuali.at p.milestone.Move-inDate		1	Move-in Date	Move-in Date for Some category of students		Admissions	Date	Sometime during the last week in August and 1st week of Sept but varies by type of student	Configured	Y	Y	
kuali.at p.milestone.NewStudentConvocation		2	New Student Convocation	New Student Convocation and Openning Exercises		Admissions	Date	Typically the day after move-in date	Configured	Y	Y	
kuali.at p.milestone.NewStudentOrientation	TRUE	3	New Student Orientation	New Student Orientation		Admissions		Typically the same day as Convocation	Configured	Y	Y	

#### Billing Milestones

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.at p.milestone.GenerateBills		1	Generate Bills	Bills Generation Date		Finance	Process Run	25th of July	Configured	Y	Y	
kuali.at p.milestone.DisburseFunds		2	Disburse Funds	Funds Disbursement Date		Finance	Process Run	18th of August	Configured	Y	Y	Why then? Why not after the term starts?
kuali.at p.milestone.PaymentDue		3	Payment Due	Payment Due Date		Finance	Deadline	One month after BillGeneration	Configured	Y	Y	
kuali.at p.milestone.ProcessRefunds		4	Process Refunds	Refund Process Date		Finance	Process Run	13 days before the first day of classes	Configured	Y	Y	

#### Financial Aid Milestones

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.at.p.milestone.FinancialAidCensus		1	Financial Aid Census	Financial Aid Census Date		Financial Aid	Process Run	6 weeks from the start of the semester	Configured	Y	Y	
kuali.at.p.milestone.BeginPacking		1	Begin Packaging	Packaging Process Begin Date		Financial Aid	Process Start	October 15th	Configured	Y	Y	Having this in October seems odd. I think KU has it wrong, packaging is normally done in the spring for the next AY

#### Scheduling Milestones

Milestone Type Key	Is a Range ?	Functional Group	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.at.p.milestone.RoomSchedulingPeriod		Scheduling		Room Scheduling	Room Scheduling for Next Year		Registrar	Open	November 5th	Placeholder	N	N	Odd that KU has Fall scheduling starting in November don't they mean Spring ?
kuali.at.p.milestone.RoomSchedulingBegin		Scheduling		Room Scheduling Begin	Rooms can begin to be scheduled			Open		Configured	Y	Y	

#### Tuition Calculation Milestones

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.at p.milestone.Refund100		1	Refund 100	Cancel w/ 100% refund Date		Finance	Deadline	First day of classes	Configured	Y	Y	
kuali.at p.milestone.Refund80		2	Refund 80	Cancel w/ 80% refund Date		Finance	Deadline	3rd week	Configured	Y	Y	
kuali.at p.milestone.Refund60		3	Refund 60	Cancel w/ 60% refund Date		Finance	Deadline	4nd week	Configured	Y	Y	
kuali.at p.milestone.Refund50		4	Refund 50	Cancel w/ 50% refund Date		Finance	Deadline	Winter term only, 2nd week of classes	Configured	Y	Y	
kuali.at p.milestone.Refund40		5	Refund 40	Cancel w/ 40% refund Date		Finance	Deadline	5th week	Configured	Y	Y	
kuali.at p.milestone.Refund20		6	Refund 20	Cancel w/ 20% refund Date		Finance	Deadline	6th week	Configured	Y	Y	

## Misc

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.at p.milestone.LeaveofAbsenceBegin			Leave of Absence Begin	Begin Leave of Absence option		Registrar	Deadline	????	Configured	Y	Y	This is also listed under "Graduation Milestones" Not sure what this was or means in KU

## Duration Types

Duration Key	Duration Name	Exp Time	Units	Duration Description	Typical Usage	Aliases	Agreed Upon?	In Constant File?	In Database ?	Comments/Questions
--------------	---------------	----------	-------	----------------------	---------------	---------	--------------	-------------------	---------------	--------------------

kuali.atp.duration.TBD	TBD	TBD	TBD	Ad hoc period required for negotiated sessions		Ad hoc	Configured - Core Slice	N	Y	Used for more experiential learning
kuali.atp.duration.TwoYears	Two Years	2	Years	Four Year Program	(2)		Configured	Y	Y	Used for program
kuali.atp.duration.FourYears	Four Years	4	Years	Four Year Program	(2)		Configured	Y	Y	Used for program
kuali.atp.duration.Month	Month	1	Month	1 month period			Configured	Y	Y	used for capturing intensity, per month
kuali.atp.duration.Week	Week	1	Week	1 week period			Configured	Y	Y	used for capturing intensity, per week
kuali.atp.duration.Day	Day	1	Day	1 day period			Configured - Core Slice	N	Y	used for capturing intensity, per day
kuali.atp.duration.Hours	Hours		Hours	1 hour period			Configured	Y	Y	Jira request to capture time for appointment windows
kuali.atp.duration.Minutes	Minutes		Minutes	1 minute period			Configured	Y	Y	Jira request to capture time for appointment windows
kuali.atp.duration.HalfSemester	Half Semester	8	Weeks	Half semester (8 weeks)	(1)		Configured	Y	Y	Not in original KU description
kuali.atp.duration.Mini-mester	Mini-mester	3	Weeks	Very Short 3 week mini-mester used for Grad in Summer	(1)		Configured	Y	Y	
kuali.atp.duration.Period	Period	4	Weeks	Short one month term used for Winter Activities	(1)		Configured	Y	Y	

kuali.atp.duration.Semester	Semester	16	Weeks	Full 16 week semester used in fall and spring			Configured	Y	Y	
kuali.atp.duration.Quarter	Quarter	16	Weeks	Full 11 week quarter			Configured - Core Slice	N	Y	
kuali.atp.duration.Session	Session	6	Weeks	Short 6 week sessions used in summer	(1)		Configured	Y	Y	
kuali.atp.duration.Term	Term	12	Weeks	12 week term used in the summer			Configured	Y	Y	
kuali.atp.duration.Year	Year	1	Year	Full Year	(2)		Configured	Y	Y	

#### Notes

1. ATPs with this durations are not allowed to be chosen to as either an effective or expiration term on a proposal for a new course.
2. This duration may not be chosen for the duration of a new course

#### Seasonal Types

Season Key	Season Name	Typically Used in	Season Description	Typical From MM/DD	Typical Thru MM/DD	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.season.AY	AY	Annual	Academic Year	1-Sep	31-Aug		Configured	Y	Y	Odd that we are putting a whole year as a season but we need something like this to allow us to rollup terms into academic years...
kuali.atp.season.FY	FY	Annual	Fiscal Year	1-Jul	30-Jun		Configured	Y	Y	
kuali.atp.season.Fall	Fall	Credit Course	Fall	1-Sep	31-Dec		Configured	Y	Y	
kuali.atp.season.Fall1	Fall 1	Credit Course	1st Half of Fall	1-Sep	15-Oct		Configured	Y	Y	
kuali.atp.season.Fall2	Fall 2	Credit Course	2nd Half of Fall	16-Oct	31-Dec		Configured	Y	Y	

kuali.atp.season.Fall-Spring	Fall-Spring	Credit Course	Fall & Spring	1-Sep	15-May		Configured	Y	Y	
kuali.atp.season.Spring	Spring	Credit Course	Spring	16-Jan	15-May		Configured	Y	Y	
kuali.atp.season.Spring1	Spring 1	Credit Course	1st Half of Spring	16-Jan	15-Mar		Configured	Y	Y	
kuali.atp.season.Spring2	Spring 2	Credit Course	2nd Half of Spring	16-Mar	15-May		Configured	Y	Y	
kuali.atp.season.SpringBreak	Spring Break	Credit Course	Spring Break	2nd week in March	2nd week in March		Configured	Y	Y	
kuali.atp.season.Summer	Summer	Credit Course	Summer	16-May	21-Aug		Configured	Y	Y	
kuali.atp.season.Summer1	Summer 1	Credit Course	1st Half of Summer	16-May	4-Jul		Configured	Y	Y	
kuali.atp.season.Summer1A	Summer 1A	Credit Course	1A of Summer	1-Jun	15-Jun		Configured	Y	Y	
kuali.atp.season.Summer1B	Summer 1B	Credit Course	1B of Summer	16-Jun	1-Jul		Configured	Y	Y	
kuali.atp.season.Summer2	Summer 2	Credit Course	2nd Half of Summer	5-Jul	21-Aug		Configured	Y	Y	
kuali.atp.season.Summer2C	Summer 2C	Credit Course	2C of Summer	1-Jul	15-Jul		Configured	Y	Y	
kuali.atp.season.Summer2D	Summer 2D	Credit Course	2D of Summer	16-Jul	1-Aug		Configured	Y	Y	
kuali.atp.season.Winter	Winter	Credit Course	Winter Activity Period	1-Jan	31-Jan		Configured	Y	Y	
kuali.atp.season.Any	Any	Lui	Any Time	Any	Any		Configured - Core Slice	N	Y	Used for more experiential learning
kuali.atp.season.FourYearCycle	Four Year Cycle	Program	Four Year Cycle				Configured	Y	Y	Used for undergraduate programs
kuali.atp.season.Term1	Term 1	Program	Term 1 of Year				Configured	Y	Y	Same as fall?

kuali.atp.season.Term2	Term 2	Program	Term 2 of Year				Configured	Y	Y	Same as spring?
kuali.atp.season.Year1	Year 1	Program	Year 1 of Program				Configured	Y	Y	Used for undergraduate programs
kuali.atp.season.Year2	Year 2	Program	Year 2 of Program				Configured	Y	Y	Used for undergraduate programs
kuali.atp.season.Year3	Year 3	Program	Year 3 of Program				Configured	Y	Y	Used for undergraduate programs
kuali.atp.season.Year4	Year 4	Program	Year 4 of Program				Configured	Y	Y	Used for undergraduate programs
kuali.atp.season.AlternateYearsCycle	Alternate Years Cycle	Offering	Every other Year Cycle				Configured	Y	Y	
kuali.atp.season.EvenYears	Even Years	Offering	Even Years				Configured	Y	Y	
kuali.atp.season.OddYears	Odd Years	Offering	Odd Years				Configured	Y	Y	

## Relationships

### ATP to ATP Types

Key	Name	Description	Example Usage	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.atp.relation.includes	Includes	The first ATP includes in the second ATP	Semesters include their associated mini-mesters , Academic Calendars include terms	Nested, Has, Contains	Configured - Core Slice	Y	N	Did not choose "contains" or "nested" because that word implies too strongly that the dates of the included ATP must be within the dates of the original ATP which is not the case.
kuali.atp.atp.relation.associated	Associated	The first ATP is associated with the second ATP	Academic Calendar are associated with Holiday Calendars		Configured - Core Slice	Y	N	Added

kuali.atp.atp.relation.precedes	Precedes	The first ATP IMMEDIATE LY precedes the second ATP	The first Quarter immediately precedes the second Quarter in a quarter based system	Before, Previous	Configured - Core Slice	Y	N	Would allow us to do term arithmetic in rules such as an Incomplete Grade must be completed by the end of the following term.  Do we need a follows or can that be inferred by the precedes.
---------------------------------	----------	----------------------------------------------------	-------------------------------------------------------------------------------------	------------------	-------------------------	---	---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## ATP to ATP States

Process Key
kuali.atp.atp.relation.lifecycle

State Key	State Name	State Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.atp.relation.state.active (initial)	Active	Indicates that this relation is active	Actual, Real	Configured	Y	Y	
kuali.atp.atp.relation.state.inactive	Inactive	Indicates that this relation is inactive	Removed	Configured	Y	Y	

## Allowed ATP to ATP Types

Type Relation Type	Seq	Name	Type Level 1	Rank	Type Level 2	Rank	Type Level 3	Rank	Type Level 3
kuali.type.type.relation.type.allowed	1	Academic Calendar can contain FallSpring	kuali.atp.type.AcademicCalendar	1	kuali.atp.type.FallSpring				
kuali.type.type.relation.type.allowed	2	Academic Calendar can contain Fall	kuali.atp.type.AcademicCalendar	2	kuali.atp.type.Fall				
kuali.type.type.relation.type.allowed	3	Fall can contain HalfFall1			kuali.atp.type.Fall	1	kuali.atp.type.HalfFall1		

kuali.type.type.relation.type.allo wed	4	Fall can contain HalfFall2			kuali.atp.type.Fall	2	kuali.atp.type.HalfFall2		
kuali.type.type.relation.type.allo wed	5	Academic Calendar can contain Winter	kuali.atp.type.AcademicCalendar	3	kuali.atp.type.Winter				
kuali.type.type.relation.type.allo wed	6	Academic Calendar can contain Spring	kuali.atp.type.AcademicCalendar	4	kuali.atp.type.Spring				
kuali.type.type.relation.type.allo wed	7	Spring can contain HalfSpring1			kuali.atp.type.Spring	1	kuali.atp.type.HalfSpring1		
kuali.type.type.relation.type.allo wed	8	Spring can contain SpringBreak			kuali.atp.type.Spring	2	kuali.atp.type.SpringBreak		
kuali.type.type.relation.type.allo wed	9	Spring can contain HalfSpring2			kuali.atp.type.Spring	3	kuali.atp.type.HalfSpring2		
kuali.type.type.relation.type.allo wed	10	Academic Calendar can contain Session1	kuali.atp.type.AcademicCalendar	5	kuali.atp.type.Summer	1	kuali.atp.type.Session1		
kuali.type.type.relation.type.allo wed	11	Session1 can contain Mini-mester1A					kuali.atp.type.Session1	1	kuali.atp.type.Mini-mester1A
kuali.type.type.relation.type.allo wed	12	Session1 can contain Mini-mester1B					kuali.atp.type.Session1	2	kuali.atp.type.Mini-mester1B
kuali.type.type.relation.type.allo wed	13	Summer can contain Session2			kuali.atp.type.Summer	2	kuali.atp.type.Session2		
kuali.type.type.relation.type.allo wed	14	Session2 can contain Mini-mester2C					kuali.atp.type.Session2	1	kuali.atp.type.Mini-mester2C
kuali.type.type.relation.type.allo wed	15	Session2 can contain Mini-mester2D					kuali.atp.type.Session2	2	kuali.atp.type.Mini-mester2D
kuali.type.type.relation.type.allo wed	16	Academic Calendar can contain SummerEve	kuali.atp.type.AcademicCalendar	6	kuali.atp.type.SummerEve				

kuali.type.type.relation.type.allo wed	17	SummerEvent can contain SessionG1			kuali.atp.type.Summe rEve	1	kuali.atp.type.Session G1		
kuali.type.type.relation.type.allo wed	18	SummerEvent can contain SessionG2			kuali.atp.type.Summe rEve	2	kuali.atp.type.Session G2		

## (CM 2.0) Academic Calendar Event Milestones

### Academic Calendar Event Milestones

kuali.milestone.type.group.acal.event -- The list of milestone types that are used for academic calendar events

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.milestone.FamilyWeekend	TRUE	1	Family Weekend	Family Weekend	Parent Weekend	University		2nd Friday in october until that Sunday	Configured - Core Slice	Y	N	
kuali.atp.milestone.Homecoming		2	Homecoming	Homecoming		University	Date	In October based on Football Schedule	Configured	Y	Y	
			Commencement	Graduation					Placeholder	N	N	
			Alumni Day						Placeholder	N	N	

⚠ This list should also include the types that are also in the Graduation Milestone group.

## (CM 2.0) Admissions Milestones

### Admissions Milestones

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.milestone.AdmissionApplicationDueEarlyCycle		1	Admissions Application Due Early Cycle	Early Admissions Application Due Date		Admissions	Deadline	December 1st	Configured	Y	Y	

Obviously there are many many more...													
---------------------------------------	--	--	--	--	--	--	--	--	--	--	--	--	--

## (CM 2.0) Allowed ATP to ATP Types

### Allowed ATP to ATP Types

Type Relation Type	Seq	Name	Type Level 1	Rank	Type Level 2	Rank	Type Level 3	Rank	Type Level 3
kuali.type.type.relation.type.allowed	1	Academic Calendar can contain FallSpring	kuali.atp.type.AcademicCalendar	1	kuali.atp.type.FallSpring				
kuali.type.type.relation.type.allowed	2	Academic Calendar can contain Fall	kuali.atp.type.AcademicCalendar	2	kuali.atp.type.Fall				
kuali.type.type.relation.type.allowed	3	Fall can contain HalfFall1			kuali.atp.type.Fall	1	kuali.atp.type.HalfFall1		
kuali.type.type.relation.type.allowed	4	Fall can contain HalfFall2			kuali.atp.type.Fall	2	kuali.atp.type.HalfFall2		
kuali.type.type.relation.type.allowed	5	Academic Calendar can contain Winter	kuali.atp.type.AcademicCalendar	3	kuali.atp.type.Winter				
kuali.type.type.relation.type.allowed	6	Academic Calendar can contain Spring	kuali.atp.type.AcademicCalendar	4	kuali.atp.type.Spring				
kuali.type.type.relation.type.allowed	7	Spring can contain HalfSpring1			kuali.atp.type.Spring	1	kuali.atp.type.HalfSpring1		
kuali.type.type.relation.type.allowed	8	Spring can contain SpringBreak			kuali.atp.type.Spring	2	kuali.atp.type.SpringBreak		
kuali.type.type.relation.type.allowed	9	Spring can contain HalfSpring2			kuali.atp.type.Spring	3	kuali.atp.type.HalfSpring2		

kuali.type.type.relation.type.allo wed	10	Academic Calendar can contain Session1	kuali.atp.type.AcademicCalendar	5	kuali.atp.type.Summe r	1	kuali.atp.type.Session 1		
kuali.type.type.relation.type.allo wed	11	Session1 can contain Mini-mester1A					kuali.atp.type.Session 1	1	kuali.atp.type.Mini-me ster1A
kuali.type.type.relation.type.allo wed	12	Session1 can contain Mini-mester1B					kuali.atp.type.Session 1	2	kuali.atp.type.Mini-me ster1B
kuali.type.type.relation.type.allo wed	13	Summer can contain Session2			kuali.atp.type.Summe r	2	kuali.atp.type.Session 2		
kuali.type.type.relation.type.allo wed	14	Session2 can contain Mini-mester2C					kuali.atp.type.Session 2	1	kuali.atp.type.Mini-me ster2C
kuali.type.type.relation.type.allo wed	15	Session2 can contain Mini-mester2D					kuali.atp.type.Session 2	2	kuali.atp.type.Mini-me ster2D
kuali.type.type.relation.type.allo wed	16	Academic Calendar can contain SummerEve	kuali.atp.type.AcademicCalendar	6	kuali.atp.type.Summe rEve				
kuali.type.type.relation.type.allo wed	17	SummerEve can contain SessionG1			kuali.atp.type.Summe rEve	1	kuali.atp.type.Session G1		
kuali.type.type.relation.type.allo wed	18	SummerEve can contain SessionG2			kuali.atp.type.Summe rEve	2	kuali.atp.type.Session G2		

## (CM 2.0) ATP Relationships

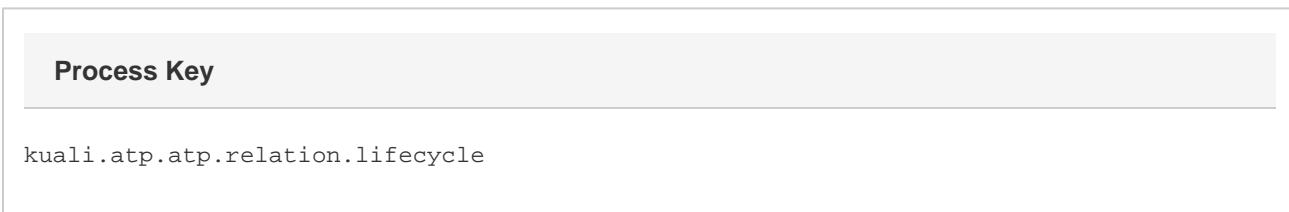
### Relationships

#### ATP to ATP Types

Key	Name	Description	Example Usage	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/ Questions

kuali.atp.atp.relation.includes	Includes	The first ATP includes in the second ATP	Semesters include their associated mini-mesters , Academic Calendars include terms	Nested, Has, Contains	Configured - Core Slice	Y	N	Did not choose "contains" or "nested" because that word implies too strongly that the dates of the included ATP must be within the dates of the original ATP which is not the case.
kuali.atp.atp.relation.associated	Associated	The first ATP is associated with the second ATP	Academic Calendar are associated with Holiday Calendars		Configured - Core Slice	Y	N	Added
kuali.atp.atp.relation.precedes	Precedes	The first ATP IMMEDIATE LY precedes the second ATP	The first Quarter immediately precedes the second Quarter in a quarter based system	Before, Previous	Configured - Core Slice	Y	N	Would allow us to do term arithmetic in rules such as an Incomplete Grade must be completed by the end of the following term.  Do we need a follows or can that be inferred by the precedes.

#### ATP to ATP States



State Key	State Name	State Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.atp.relation.state.active (initial)	Active	Indicates that this relation is active	Actual, Real	Configured	Y	Y	
kuali.atp.atp.relation.state.inactive	Inactive	Indicates that this relation is inactive	Removed	Configured	Y	Y	

#### (CM 2.0) ATP States

## ATP States

Process Key							
kuali.atp.lifecycle							

State Key	State Name	State Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.state.Draft (initial)	Draft	Indicates that this Time Period is just tentative	Planned, Proposed, Tentative	Configured	Y	Y	 do we need/want a "rolled over" state or is draft good enough?
kuali.atp.state.Official	Official	Indicates that this Time Period has been established	Actual, Real, Publishable	Configured	Y	Y	There is no state to indicate that this is the current one. That is determined by the dates. This is still the Official academic calendar or term for the period to which it applies

## (CM 2.0) Billing Milestones

### Billing Milestones

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.milestone.GenerateBills		1	Generate Bills	Bills Generation Date		Finance	Process Run	25th of July	Configured	Y	Y	
kuali.atp.milestone.DisburseFunds		2	Disburse Funds	Funds Disbursement Date		Finance	Process Run	18th of August	Configured	Y	Y	Why then? Why not after the term starts?
kuali.atp.milestone.PaymentDue		3	Payment Due	Payment Due Date		Finance	Deadline	One month after BillGeneration	Configured	Y	Y	

kuali.atp.milestone.ProcessRefunds		4	Process Refunds	Refund Process Date		Finance	Process Run	13 days before the first day of classes	Configured	Y	Y	
------------------------------------	--	---	-----------------	---------------------	--	---------	-------------	-----------------------------------------	------------	---	---	--

## (CM 2.0) Calendar Types

### Calendar Types

ATP Type Key	ATP Type Name	ATP Type Description	Typical Usage	Season Type	Duration Type	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.type.AcademicCalendar	Academic Calendar	Academic Calendar	Annual	AY	Year	Program Calendar	Configured	Y	Y	<p>💡 How do we support an academic calendar for an 18 month program?</p> <p>==&gt; Perhaps an Academic Calendar Type can be associated with more than one Atp Type (similar to Credential Program can map to bachelors or masters or doctorate credential types)... if so we need a grouping .</p>
kuali.atp.type.HolidayCalendar	Holiday Calendar	Holiday Calendar	Annual	AY	Year		Configured	Y	Y	

## (CM 2.0) Curriculum Milestones

### Curriculum Milestones

kuali.milestone.type.group.curriculum-- The list of milestone types that are used to drive the curriculum management process

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.at.p.milestone.C oordin atorsKi ckoffMeeti ng		1	Coordin ators Kickoff Meetin g	Coordin ators Kickoff Meetin g		Registr ar	Meetin g	1st of Octob er	Config ured	Y	Y	
kuali.at.p.miles tone.C urricul umCom mitteeMeeti ng		2	Curric ulum Commi ttee Meetin g	Curric ulum Commi ttee Meetin g	after the deadline for proposals. Thereafter weekly every Tuesday, skipping for Christmas break, until the week after the final review period in April.	Registr ar	Meetin g	First meetin g on the Tuesday the week	Config ured	Y	Y	
kuali.at.p.milestone.P roposalPeriod	TRUE	3	Propos al Period	Propos al Period		Registr ar		Middle of Septe mber to the end of the next August	Config ured - Core Slice	Y	N	It used to be tighter!!!!!! Used as Date Range
kuali.at.p.milestone.M ajorCh anges Deadli ne		4	Major Chang es Deadli ne	Deadli ne for Propos als with Major Chang es		Registr ar	Deadli ne	The same day as Final Exams for the fall term	Config ured	Y	Y	

kuali.atp.miles tone.M inorCh anges Deadli ne		5	Minor Chang es Deadli ne	Deadli ne for Propos als with Minor Chang es		Registr ar	Deadli ne	Aroun d the 10th of Januar y	Config ured	Y	Y	
kuali.atp.miles tone.L astMin utePro posals Deadli ne		6	Last Minute Propos als Deadli ne	Deadli ne for Last Minute Propos als		Registr ar	Deadli ne	Two weeks before changes need to be published on-line for that term	Config ured	Y	Y	
kuali.atp.miles tone.R eview Period	TRUE	7	Revie w Period	Period within which propos al are group ed and review ed together for approv al		Registr ar		3 of these in a year, one from Oct to the middle of februar y, 2nd one from the middle of februar y to the the 3rd week in March, and the final one from that period until the start of the next fall term	Config ured - Core Slice	Y	N	Used as Date Range
kuali.atp.miles tone.P ublish Chang esOnLi ne		8	Publis h Chang es On Line	Approved Cours e Chang es Publis hed On-Lin e		Registr ar	Proces s	The day before Advance registr ation opens	Config ured	Y	Y	Did not exist in KU but need something like them

## (CM 2.0) Duration Types

### Duration Types

Duration Key	Duration Name	Exp Time	Units	Duration Description	Typical Usage	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.duration.TBD	TBD	TBD	TBD	Ad hoc period required for negotiated sessions		Ad hoc	Configured - Core Slice	N	Y	Used for more experiential learning
kuali.atp.duration.TwoYears	Two Years	2	Years	Four Year Program	(2)		Configured	Y	Y	Used for program
kuali.atp.duration.FourYears	Four Years	4	Years	Four Year Program	(2)		Configured	Y	Y	Used for program
kuali.atp.duration.Month	Month	1	Month	1 month period			Configured	Y	Y	used for capturing intensity, per month
kuali.atp.duration.Week	Week	1	Week	1 week period			Configured	Y	Y	used for capturing intensity, per week
kuali.atp.duration.Day	Day	1	Day	1 day period			Configured - Core Slice	N	Y	used for capturing intensity, per day
kuali.atp.duration.Hours	Hours		Hours	1 hour period			Configured	Y	Y	Jira request to capture time for appointment windows
kuali.atp.duration.Minutes	Minutes		Minutes	1 minute period			Configured	Y	Y	Jira request to capture time for appointment windows
kuali.atp.duration.HalfSemester	Half Semester	8	Weeks	Half semester (8 weeks)	(1)		Configured	Y	Y	Not in original KU description
kuali.atp.duration.Mini-mester	Mini-mester	3	Weeks	Very Short 3 week mini-mester used for Grad in Summer	(1)		Configured	Y	Y	

kuali.atp.duration.Period	Period	4	Weeks	Short one month term used for Winter Activities	(1)		Configured	Y	Y	
kuali.atp.duration.Semester	Semester	16	Weeks	Full 16 week semester used in fall and spring			Configured	Y	Y	
kuali.atp.duration.Quarter	Quarter	16	Weeks	Full 11 week quarter			Configured - Core Slice	N	Y	
kuali.atp.duration.Session	Session	6	Weeks	Short 6 week sessions used in summer	(1)		Configured	Y	Y	
kuali.atp.duration.Term	Term	12	Weeks	12 week term used in the summer			Configured	Y	Y	
kuali.atp.duration.Year	Year	1	Year	Full Year	(2)		Configured	Y	Y	

#### Notes

1. ATPs with this durations are not allowed to be chosen to as either an effective or expiration term on a proposal for a new course.
2. This duration may not be chosen for the duration of a new course

## (CM 2.0) Dynamic Attribute Types

### *Dynamic Attribute Types*

Dynamic Attribute Type Key	Type Name	Type Description	Typical Usage	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.attribute.campus.key	Campus Key	Holds keys that indicate associated campuses	On Holiday Calendars	Location	Configured - Core Slice	N	N	

## (CM 2.0) Financial Aid Milestones

### *Financial Aid Milestones*

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.milestone.FinancialAidCensus		1	Financial Aid Census	Financial Aid Census Date		Financial Aid	Process Run	6 weeks from the start of the semester	Configured	Y	Y	

kuali.at.p.milestone.BeingPacking		1	Begin Packaging	Packaging Process Begin Date		Financial Aid	Process Start	October 15th	Configured	Y	Y	Having this in October seems odd. I think KU has it wrong, packaging is normally done in the spring for the next AY
-----------------------------------	--	---	-----------------	------------------------------	--	---------------	---------------	--------------	------------	---	---	---------------------------------------------------------------------------------------------------------------------

## (CM 2.0) Graduation Milestones

### *Graduation Milestones*

kuali.milestone.type.group.acal.event.graduation -- The list of milestone types that are typically used for non-instructional holidays

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.at.p.milestone.GraduationApplicationDeadline		1	Graduation Application Deadline	Deadline to apply for Graduation		University	Deadline	Nov 15th for the fall, Summer August 1st	Configured	Y	Y	KU has no deadline for Spring which is unusual I think it is missing
kuali.at.p.milestone.AlumniDay		1	Alumni Day	Alumni Day	Welcome back day	University	Date	2nd Saturday in May	Configured	Y	Y	Also an Event?
kuali.at.p.milestone.Baccalaureate		2	Baccalaureate	Baccalaureate		University	Date	The day after alumni day	Configured	Y	Y	
kuali.at.p.milestone.Commencement		3	Commencement	Commencement		University	Date	They day after Baccalaureate	Configured	Y	Y	
kuali.at.p.milestone.LeaveofAbsenceBegin		4	Leave of Absence Begin	Leave of Absence Begin			Date		Configured	Y	Y	

## (CM 2.0) Holiday Milestones

### Holidays

```
kuali.milestone.type.group.holidays -- The list of milestone types that are allowed for holidays
```

⚠ The list should include both the instructional holidays and the non-instructional holidays below.

#### Non-Instructional Holidays

```
kuali.milestone.type.group.holidays.non-instructional -- The list of milestone types that are typically used for non-instructional holidays
```

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.at.p.milestone.LaborDay		1	Labor Day	Labor Day		University	Date	1st Monday in September	Configured	Y	Y	
kuali.at.p.milestone.VeteransDay		3	Veterans Day	Veterans Day		University	Date	November 11th.	Configured - Core Slice	N	Y	Added 1/10 TL
kuali.at.p.milestone.VeteransDayObserved		3	Veterans Day Observed	Veterans Day Observed		University	Date	November 11th. If holiday falls on Saturday, observed on Friday. If holiday falls on Sunday, observed on Monday.	Configured - Core Slice	N	Y	Added 1/10 TL
kuali.at.p.milestone.ThanksgivingBreak	TRUE	4	Thanks giving Break	Thanks giving Break		University	Open	4th Thursday-Sunday of November	Configured	Y	Y	Changed from Wed-Mon. 1/10 TL
kuali.at.p.milestone.Christmas		5	Christm as Day	Christm as Day		University	Date	December 25th.	Configured - Core Slice	N	Y	Added 1/10 TL

kuali.at p.milest one.Ch ristmas Observ ed		5	Christm as Day Observ ed	Christm as Day Observ ed		Univers ity	Date	Decem ber 25th. If holiday falls on Saturda y, observ ed on Friday. If holiday falls on Sunday ,, observ ed on Monda y.	Configu red - Core Slice	N	Y	Added 1/10 TL
kuali.at p.milest one.Ne wYears Day		6	New Years Day	New Years Day		Univers ity	Date	January 1.	Configu red - Core Slice	N	Y	Added 1/10 TL
kuali.at p.milest one.Ne wYears DayOb served		6	New Years Day Observ ed	New Years Day Observ ed		Univers ity	Date	January 1. If holiday falls on Saturda y, observ ed on Friday. If holiday falls on Sunday ,, observ ed on Monda y.	Configu red - Core Slice	N	Y	Added 1/10 TL
kuali.at p.milest one.ML KDay		7	MLK Day	Martin Luther King, Jr. Day		Univers ity	Date	3rd Monday in January	Placeh older	N	N	Was MLKDa yObser ved. 1/10 TL
kuali.at p.milest one.ML KDayO bserved			MLK Day Observ ed	Martin Luther King Jr. Day Observ ed		Univers ity	Date	3rd Monday in January	Configu red	Y	Y	
kuali.at p.milest one.Pre sidents Day		8	Preside nts Day	Preside nts Day		Univers ity	Date	3rd Monday in Februar y	Configu red - Core Slice	N	Y	Added 1/10 TL
kuali.at p.milest one.Me morialD ay		10	Memori al Day	Memori al Day		Univers ity	Date	Last Monday in May	Configu red - Core Slice	N	Y	Was Memori alDayO bserve d. 1/10 TL

kuali.at.p.milestone.MemorialDayObserved			Memorial Day Observed	Memorial Day Observed		University	Date	Last Monday in May	Configured	Y	Y	
kuali.at.p.milestone.IndependenceDay		11	Independence Day	Independence Day		University	Date	July 4th.	Configured - Core Slice	N	Y	
kuali.at.p.milestone.IndependenceDayObserved		11	Independence Day Observed	Independence Day Observed		University	Date	July 4th. If holiday falls on Saturday, observed on Friday. If holiday falls on Sunday, observed on Monday.	Configured	Y	Y	
kuali.at.p.milestone.OtherNonInstructionalHoliday		12	Other Non-Instructional Holiday Day	Other Holiday Managed as Non-Instructional		University	Date		Placeholder	N	N	
kuali.at.p.milestone.OtherNonInstructionalDay		1	Other non-Instructional Day	A day on which instruction does not occur but is not typically considered a holiday					Placeholder	N	N	

#### Instructional Holidays

kuali.milestone.type.group.holidays.instructional -- The list of milestone types that are typically used for instructional holidays

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
--------------------	--------------	------	-----------	-------------	---------	--------------------------	-----------	-------	--------------	-------------------	--------------	--------------------

kuali.at.p.milestone.YomKippur	??		Yom Kippur	Yom Kippur		Date		Example	N	N	UW: Occasionally Non-Instructional
kuali.at.p.milestone.GoodFriday			Good Friday	Good Friday		Date	Friday before Easter	Example	N	N	
kuali.at.p.milestone.FallBreak	TRUE		Fall Break	Fall Break	University	Open	Columbus Day Weekend	Configured - Core Slice	Y	N	Used as a Date Range N/A for UW: Moved from non-instr. 1/10 TL
kuali.at.p.milestone.SpringBreak	True		Spring Break	Spring Break Begins	University	Open	Friday of the first full week in March and classes resume Monday following the week break.	Configured - Core Slice	Y	N	Used as a Date Range N/A for UW: Moved from non-instr. 1/10 TL
kuali.at.p.milestone.OtherInstructionalHoliday			Other Instructional Holiday	Other Holiday Managed as Instructional				Placeholder	N	N	Added 1/10 TL
kuali.at.p.milestone.ColumbusDay			Columbus Day	Columbus Day	University	Date		Configured	Y	Y	<input checked="" type="checkbox"/> 803 - Co "Approve Instruction in Const. Close <input checked="" type="checkbox"/> 805 - Co "Approve Instruction in Database Close UMD Ref Data

kuali.at p.milestone.ele ctionday			Election Day	Election Day		Univers ity	Date		Configu red	Y	Y	<a href="#">803 - Co "Approve Instruction in Constat Closed"</a>
kuali.at p.milestone.ros hshash nah			Rosh Hashan ah	Rosh Hashan ah		Univers ity	Date		Configu red	Y	Y	<a href="#">803 - Co "Approve Instruction in Constat Closed"</a>

## (CM 2.0) Key Date Milestones

### Key Date Milestones

kuali.milestone.type.group.keydate-- The list of milestone types that are used for Key Dates that are associated with a term.

Mile stone Type Key	Hard ened @ Clas s II Level ?	Is a Rang e?	Sort	Type Name	Desc riptio n	Alias es	Respo nsible Orga nizati on	How used ?	Calc ulati on Rule s	Agre ed Upon ?	In Cons tant File?	In Data base ?	Com ment s/Qu estio ns	★ M aps to Refer ence Insti tution	★ N otes re: Refer ence Insti tution

kuali.atp.milestone.AdvancedRegistrationPeriod		TRUE	1	Advance Registration Period	Advance Registration Period	Preeregistration period	Registrar		First two weeks in November for Spring, Last week in March to 1st week in April for Fall and Summer. Fall and Summer are the same time period	Configured - Core Slice	Y	N		Y	Used as a Date Range UW has TWO "Advance" Periods - Periods 1 and 2	
kuali.atp.milestone.RegistrationPeriod	TRUE	TRUE	2	Registration Period	Registration Period	Course Selection Period	Registrar		Typically the first week of the semester	Configured - Core Slice	Y	N	?	Is there difference between the end of the registration period and the Course Selection Period End Date below?	Y	UW Period 3

kuali.atp.milestone.Registration.Period1			Registration Period 1	Freshmen		Registrar		Configured	Y	Y	<b>KSENROLL-2</b> <b>806</b> - Configure "Approved" Key Dates in Constants File (Closed)	<b>KSENROLL-2</b> <b>807</b> - Configure "Approved" Key Dates in Database (Closed)	Implementing Institution Needs
kuali.atp.milestone.Registration.Period2			Registration Period 2	Sophomore		Registrar		Configured	Y	Y	<b>KSENROLL-2</b> <b>806</b> - Configure "Approved" Key Dates in Constants File (Closed)	<b>KSENROLL-2</b> <b>807</b> - Configure "Approved" Key Dates in Database (Closed)	Implementing Institution Needs
kuali.atp.milestone.Registration.Period3			Registration Period 3	Juniors		Registrar		Configured	Y	Y	<b>KSENROLL-2</b> <b>806</b> - Configure "Approved" Key Dates in Constants File (Closed)	<b>KSENROLL-2</b> <b>807</b> - Configure "Approved" Key Dates in Database (Closed)	Implementing Institution Needs

kuali.atp.milestone.Registration.Period4				Registration Period 4	Seniors		Registrar		Configured	Y	Y	<input checked="" type="checkbox"/> <b>KSENROLL-2</b> <b>806</b> - Configure "Approved" Key Dates in Constants File (Closed)	
												<input checked="" type="checkbox"/> <b>KSENROLL-2</b> <b>807</b> - Configure "Approved" Key Dates in Database (Closed)	
				Late Registration Period 1					Configured	Y	Y	Implementing Institution Needs	
kuali.atp.milestone.Registration.Period5				Late Registration Period 1					Configured	Y	Y	Dan E. found these in the Data base	
kuali.atp.milestone.Registration.Period6				Late Registration Period 2					Configured	Y	Y	Dan E. found these in the Data base	
kuali.atp.milestone.Registration.Period7				Schedule Changes					Configured	Y	Y	Dan E. found these in the Data base	
kuali.atp.milestone.Registration.BeginsforMBA		2.1	Registration Begins for MBA	MBA Registration begins		Registrar	Open	Fall only the last week in August	Configured	Y	Y	Is this really "registration period" but on the MBA calendar?	
kuali.atp.milestone.Registration.BeginsNonDegreree		2.2	Registration Begins Non Degreree	Registration for non-degree seeking students		Registrar	Open	Fall only 1st of September	Configured	Y	Y		

kuali.atp.milestone.Registration BeginsTransfer			2.3	Registration Begins Transfer	Registration for Transfer Students		Registar	Open	Typically the day after move-in date	Configured	Y	Y			
kuali.atp.milestone.Instructional Period	TRUE	TRUE	3	Instructional Period	Dates between which classes should meet	Classes begin , and end dates First meeting date and last meeting date of term	Registar	Open	For Fall the day after labor day, for Spring the day after MLK day and the last day For fall the Friday the week before Christmas, for Spring, the Friday the week before Memorial Day	Configured	Y	Y		Yes	
kuali.atp.milestone.Course SelectionPeriod End	TRUE		4	Course Selection Period	Course Selection Period ends	Last date to Add Date	Registar	Deadline	2 weeks into the term	Configured	Y	Y	Is this what I've called Add Date ? Is this the same as the end date of the "registration period"?	Yes	

kuali.atp.milestone.DropDeadlineWithoutRecord			5	Drop Deadline Without Record	Deadline to drop w/o "W"	Late Drop Date	Registar	Deadline	Only applies to Summer and Winter terms, set approximately halfway through the term.	Configured	Y	Y		Yes	
kuali.atp.milestone.DropDate	TRUE		6	Drop Date	Drop Period ends	Last date to withdraw from a class	Registar	Deadline	The Friday 6 weeks from the start of the Semester, only applies to Semesters not Summer and Winter	Configured	Y	Y			
kuali.atp.milestone.PostGradesMidterm			7	Post Grades Midterm	Mid-term Grades Posted		Registar	Process Run	15th of October for fall.	Configured	Y	Y	KU does not specify midterm grades for spring. Was this intentional?		
kuali.atp.milestone.MailProgressReports			8	Mail Progress Reports	Satisfactor y Progress Reports/Letters		Registar	Process Run	The same day the midterms grades are due	Configured	Y	Y	KU actually did not specify this date		

kuali.atp.milestone.Readin gPeriod		TRUE	10	Reading Period	Reading Days	Final Exam Preparation, Study Period	Registrar		Fall starting on the Wednesday two weeks before Christmas, for Spring the first full week in May	Configured	Y	Y			
kuali.atp.milestone.FinalExamPeriod	TRUE	TRUE	11	Final Exam Period	Final Examinations		Registrar		Based on Football Schedule	Configured	Y	Y		Yes	Currently only the Last Day of the Period is recorded
kuali.atp.milestone.GradesDue	TRUE		12	Grades Due	Grades Due		Registrar	Deadline	July 7 for Summer session	Configured	Y	Y	?	is there a grading Period? i.e a start date?	
kuali.atp.milestone.PostGrades			13	Post Grades	Grades Poste d	Publi sh Grades	Registrar	Process Run	Fall the day after Christmas, Spring the last Friday before Commencement, for the winter term the 10th of february	Configured	Y	Y			

## (CM 2.0) Milestone States

### Milestone States

Process Key								
kuali.milestone.lifecycle								

State Key	State Name	State Description	Aliases	Status	In Constant File?	In Database?	Comments/Questions
kuali.milestone.state.Draft (initial)	Draft	Indicates that this Milestone is just tentative	Planned, Proposed, Tentative	Configured	Y	Y	
kuali.milestone.state.Official	Official	Indicates that this Milestone has been established	Actual, Real	Configured	Y	Y	

## (CM 2.0) Misc Milestones

### Misc

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.milestone.LeaveofAbsenceBegin			Leave of Absence Begin	Begin Leave of Absence option		Registrar	Deadline	????	Configured	Y	Y	This is also listed under "Graduation Milestones" Not sure what this was or means in KU

## (CM 2.0) Orientation Milestones

### Orientation Milestones

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions

kuali.atp.milestone.Move-inDate		1	Move-in Date	Move-in Date for Some category of students		Admissions	Date	Sometime during the last week in August and 1st week of Sept but varies by type of student	Configured	Y	Y	
kuali.atp.milestone.NewStudentConvocation		2	New Student Convocation	New Student Convocation and Openning Exercises		Admissions	Date	Typically the day after move-in date	Configured	Y	Y	
kuali.atp.milestone.NewStudentOrientation	TRUE	3	New Student Orientation	New Student Orientation		Admissions		Typically the same day as Convocation	Configured	Y	Y	

## (CM 2.0) Program Related Types

### Program Related Types

ATP Type Key	ATP Type Name	ATP Type Description	Typical Usage	Season Type	Duration Type	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.type.UndergradProgram	Undergrad Program	Four Year Undergraduate	Program	Four Year Cycle	Four Years		Configured	Y	Y	
kuali.atp.type.FreshmanYear	Freshman Year	Freshman Year	Program	Year 1	Year		Configured	Y	Y	
kuali.atp.type.FreshmanYearTerm1	Freshman Year Term 1	1st Term Freshman	Program	Term 1	Semester		Configured	Y	Y	
kuali.atp.type.FreshmanYearTerm2	Freshman Year Term 2	2nd Term Freshman	Program	Term 2	Semester		Configured	Y	Y	
kuali.atp.type.SophomoreYear	Sophomore Year	Sophomore Year	Program	Year 2	Year		Configured	Y	Y	
kuali.atp.type.SophomoreYearTerm1	Sophomore Year Term 1	1st Term Sophomore	Program	Term 1	Semester		Configured	Y	Y	

kuali.atp.type.SophomoreYearTerm2	Sophomore Year Term 2	2nd Term Sophomore	Program	Term 2	Semester		Configured	Y	Y	
kuali.atp.type.JuniorYear	Junior Year	Junior Year	Program	Year 1	Year		Configured	Y	Y	
kuali.atp.type.JuniorYearTerm1	Junior Year Term 1	1st Term Junior	Program	Term 1	Semester		Configured	Y	Y	
kuali.atp.type.JuniorYearTerm2	Junior Year Term 2	2nd Term Junior	Program	Term 2	Semester		Configured	Y	Y	
kuali.atp.type.SeniorYear	Senior Year	Senior Year	Program	Year 4	Year		Configured	Y	Y	
kuali.atp.type.SeniorYearTerm1	Senior Year Term 1	1st Term Senior	Program	Term 1	Semester		Configured	Y	Y	
kuali.atp.type.SeniorYearTerm2	Senior Year Term 2	2nd Term Senior	Program	Term 2	Semester		Configured	Y	Y	

## (CM 2.0) Reporting Types

### Reporting Types

ATP Type Key	ATP Type Name	ATP Type Description	Typical Usage	Season Type	Duration Type	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.type.AY	AY	Full Academic Year	Annual	AY	Year		Configured	Y	Y	At present only used to group terms for display and reporting, not for LUI creation
kuali.atp.type.FY	FY	Fiscal Year	Annual	FY	Year		Configured	Y	Y	At present only used to group terms for display and reporting, not for LUI creation

## (CM 2.0) Scheduling Milestones

### Scheduling Milestones

Milestone Type Key	Is a Range ?	Functional Group	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.milestone.RoomSchedulingPeriod		Scheduling		Room Scheduling	Room Scheduling for Next Year		Registrar	Open	November 5th	Placeholder	N	N	Odd that KU has Fall scheduling starting in November don't they mean Spring ?
kuali.atp.milestone.RoomSchedulingBegin		Scheduling		Room Scheduling Begin	Rooms can begin to be scheduled			Open		Configured	Y	Y	

## (CM 2.0) Seasonal Types

### Seasonal Types

Season Key	Season Name	Typically Used in	Season Description	Typical From MM/DD	Typical Thru MM/DD	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.season.AY	AY	Annual	Academic Year	1-Sep	31-Aug		Configured	Y	Y	Odd that we are putting a whole year as a season but we need something like this to allow us to rollup terms into academic years...
kuali.atp.season.FY	FY	Annual	Fiscal Year	1-Jul	30-Jun		Configured	Y	Y	
kuali.atp.season.Fall	Fall	Credit Course	Fall	1-Sep	31-Dec		Configured	Y	Y	
kuali.atp.season.Fall1	Fall 1	Credit Course	1st Half of Fall	1-Sep	15-Oct		Configured	Y	Y	

kuali.atp.season.Fall2	Fall 2	Credit Course	2nd Half of Fall	16-Oct	31-Dec		Configured	Y	Y	
kuali.atp.season.Fall-Spring	Fall-Spring	Credit Course	Fall & Spring	1-Sep	15-May		Configured	Y	Y	
kuali.atp.season.Spring	Spring	Credit Course	Spring	16-Jan	15-May		Configured	Y	Y	
kuali.atp.season.Spring1	Spring 1	Credit Course	1st Half of Spring	16-Jan	15-Mar		Configured	Y	Y	
kuali.atp.season.Spring2	Spring 2	Credit Course	2nd Half of Spring	16-Mar	15-May		Configured	Y	Y	
kuali.atp.season.SpringBreak	Spring Break	Credit Course	Spring Break	2nd week in March	2nd week in March		Configured	Y	Y	
kuali.atp.season.Summer	Summer	Credit Course	Summer	16-May	21-Aug		Configured	Y	Y	
kuali.atp.season.Summer1	Summer 1	Credit Course	1st Half of Summer	16-May	4-Jul		Configured	Y	Y	
kuali.atp.season.Summer1A	Summer 1A	Credit Course	1A of Summer	1-Jun	15-Jun		Configured	Y	Y	
kuali.atp.season.Summer1B	Summer 1B	Credit Course	1B of Summer	16-Jun	1-Jul		Configured	Y	Y	
kuali.atp.season.Summer2	Summer 2	Credit Course	2nd Half of Summer	5-Jul	21-Aug		Configured	Y	Y	
kuali.atp.season.Summer2C	Summer 2C	Credit Course	2C of Summer	1-Jul	15-Jul		Configured	Y	Y	
kuali.atp.season.Summer2D	Summer 2D	Credit Course	2D of Summer	16-Jul	1-Aug		Configured	Y	Y	
kuali.atp.season.Winter	Winter	Credit Course	Winter Activity Period	1-Jan	31-Jan		Configured	Y	Y	
kuali.atp.season.Any	Any	Lui	Any Time	Any	Any		Configured - Core Slice	N	Y	Used for more experiential learning
kuali.atp.season.FourYearCycle	Four Year Cycle	Program	Four Year Cycle				Configured	Y	Y	Used for undergraduate programs

kuali.atp.season.Term1	Term 1	Program	Term 1 of Year				Configured	Y	Y	Same as fall?
kuali.atp.season.Term2	Term 2	Program	Term 2 of Year				Configured	Y	Y	Same as spring?
kuali.atp.season.Year1	Year 1	Program	Year 1 of Program				Configured	Y	Y	Used for undergraduate programs
kuali.atp.season.Year2	Year 2	Program	Year 2 of Program				Configured	Y	Y	Used for undergraduate programs
kuali.atp.season.Year3	Year 3	Program	Year 3 of Program				Configured	Y	Y	Used for undergraduate programs
kuali.atp.season.Year4	Year 4	Program	Year 4 of Program				Configured	Y	Y	Used for undergraduate programs
kuali.atp.season.AlternateYearsCycle	Alternate Years Cycle	Offering	Every other Year Cycle				Configured	Y	Y	
kuali.atp.season.EvenYears	Even Years	Offering	Even Years				Configured	Y	Y	
kuali.atp.season.OddYears	Odd Years	Offering	Odd Years				Configured	Y	Y	

## (CM 2.0) Seatpool Milestones

### Seatpool Milestones

kuali.milestone.type.group.seatpool-- The list of milestone types that are used to drive the seatpool management process

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How Used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.milestone.firstdayofclasses			First Day of Classes						Configured	Y	Y	
kuali.atp.milestone.lastdayofforegistration			Last Day of Registration						Configured	Y	Y	

kuali.at.p.milestone.en.doffirst.weekof.classes			End of First Week of Classes							Configured	Y	Y	
kuali.at.p.milestone.monthprior.tostartof.classes			Month Prior to Start of Classes							Configured	Y	Y	

## (CM 2.0) Term Types Grouping

### *Term Types Grouping*

kuali.atp.type.group.term -- The list of ATP types that are considered terms in that courses may be offered during them.

ATP Type Key	Sub-Term of what term?	ATP Type Name	ATP Type Description	Typical Usage	Season Type	Duration Type	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions	★ Maps to Reference Institution	★ Notes re: Reference Institution
kuali.at.p.type.Fall		Fall	Fall Semester	Credit Course	Fall	Semester		Configured	Y	Y		Yes	Duration Type: Quarter Season: Autumn Name: Autumn Quarter
kuali.at.p.type.FallSpring		Fall-Spring	Fall & Spring Semesters	Credit Course	Fall	2 Semesters		Placeholder	N	N	Used for those year long courses that start in the fall and end in the spring		

kuali.at p.type. HalfFall1	Fall	Half Fall 1	1st Half Semester in Fall	Credit Course	Fall 1	Half Semester		Configured	Y	Y			
kuali.at p.type. HalfFall2	Fall	Half Fall 2	2nd Half Semester in Fall	Credit Course	Fall 2	Half Semester		Configured	Y	Y			
kuali.at p.type. Winter		Winter	Winter Activity Period	Credit Course	Winter	Period		Configured	Y	Y		Yes	Duration Type: Quarter Season: Winter Name:  Winter Quarter
kuali.at p.type. Spring		Spring	Spring Semester	Credit Course	Spring	Semester		Configured	Y	Y		Yes	Duration Type: Quarter Season: Spring Name: Spring Quarter
kuali.at p.type. HalfSpring1	Spring	Half Spring 1	1st Half Semester in Spring	Credit Course	Spring 1	Half Semester		Configured	Y	Y			
kuali.at p.type. HalfSpring2	Spring	Half Spring 2	2nd Half Semester in Spring	Credit Course	Spring 2	Half Semester		Configured	Y	Y			

kuali.at p.type. Spring Break	Spring	Spring Break	Spring Break Experi ential Term	Credit Cours e	Spring Break	Week		Config ured	Y	Y	Do not confus e this with the Milesto ne that define s the Spring Break period where typicall y no classe s are held. This is actuall y an examp le of a term that can be used for special week long experi ential or service learnin g progra ms that over spring break.	
kuali.at p.type. Summ er		Summ er	Summ er	Credit Cours e	Summ er	Term		Config ured	Y	Y	Used to group periод s for display and reporti ng	Yes  Duration: Type: Quarter: Season: Summer Name: Summer Quarter

kuali.at p.type. Summ erEve	Summ er	Summ er Eve	Summ er Evenin g	Credit Cours e	Summ er	Term		Config ured	Y	Y	Used to group peri ods for display and reporti ng and for LUI creatio n only for evenin g classe s		
kuali.at p.type. Sessio n1	Summ er	Sessio n 1	1st Summ er Sessio n	Credit Cours e	Summ er 1	Sessio n		Config ured	Y	Y		Yes	Duration Type: Subter m Seaso n: Summ er Name: Summ er, A-Ter m Notes: Runs concur rent with full Summ er Quarte r, NO overla p with B-Ter m
kuali.at p.type. Mini-m ester1 A	Sessio n 1	Mini-m ester 1A	Summ er Mini-m ester 1A	Credit Cours e	Summ er 1A	Mini-m ester		Config ured	Y	Y			
kuali.at p.type. Mini-m ester1 B	Sessio n 1	Mini-m ester 1B	Summ er Mini-m ester 1B	Credit Cours e	Summ er 1B	Mini-m ester		Config ured	Y	Y			

kuali.at p.type. Session2	Summ er	Sessio n 2	2nd Summ er Sessio n	Credit Cours e	Summ er 2	Sessio n		Config ured	Y	Y		Yes	Durati on Type: Subter m Seaso n: Summ er Name: Summ er, B-Ter m Notes: Runs concur rent with full Summ er Quarte r, NO overla p with A-Ter m
kuali.at p.type. Mini-m ester2C	Sessio n 2	Mini-m ester 2C	Summ er Mini-m ester 2C	Credit Cours e	Summ er 2C	Mini-m ester		Config ured	Y	Y			
kuali.at p.type. Mini-m ester2D	Sessio n 2	Mini-m ester 2D	Summ er Mini-m ester 2D	Credit Cours e	Summ er 2D	Mini-m ester		Config ured	Y	Y			
kuali.at p.type. SessionG1	Summ er	Sessio n G1	1st Grad Summ er Sessio n	Credit Cours e	Summ er 1	Sessio n		Config ured	Y	Y			
kuali.at p.type. SessionG2	Summ er	Sessio n G2	2nd Grad Summ er Sessio n	Credit Cours e	Summ er 2	Sessio n		Config ured	Y	Y			
kuali.at p.type. Adhoc		Adhoc	Ad hoc sessio n	Lui	Any	TBD	On-off	Placeh older	N	N		Used for one-off course s that are taught on their own sched ule	

## (CM 2.0) Tuition Calculation Milestones

## Tuition Calculation Milestones

Milestone Type Key	Is a Range ?	Sort	Type Name	Description	Aliases	Responsible Organization	How used?	Rules	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.at.p.milestone.Refund100		1	Refund 100	Cancel w/ 100% refund Date		Finance	Deadline	First day of classes	Configured	Y	Y	
kuali.at.p.milestone.Refund80		2	Refund 80	Cancel w/ 80% refund Date		Finance	Deadline	3rd week	Configured	Y	Y	
kuali.at.p.milestone.Refund60		3	Refund 60	Cancel w/ 60% refund Date		Finance	Deadline	4nd week	Configured	Y	Y	
kuali.at.p.milestone.Refund50		4	Refund 50	Cancel w/ 50% refund Date		Finance	Deadline	Winter term only, 2nd week of classes	Configured	Y	Y	
kuali.at.p.milestone.Refund40		5	Refund 40	Cancel w/ 40% refund Date		Finance	Deadline	5th week	Configured	Y	Y	
kuali.at.p.milestone.Refund20		6	Refund 20	Cancel w/ 20% refund Date		Finance	Deadline	6th week	Configured	Y	Y	

## (CM 2.0) When Course Offered Types

### When Course Offered Types

ATP Type Key	ATP Type Name	ATP Type Description	Typical Usage	Season Type	Duration Type	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.atp.type.Even Years	Even Years	Even Numbered Years	Offering	Even Years	Year		Configured	Y	Y	
kuali.atp.type.FallEvenYears	Fall Even Years	Fall Even Years	Credit Course	Fall	Semester		Configured	Y	Y	
kuali.atp.type.SpringEvenYears	Spring Even Years	Spring Even Years	Credit Course	Spring	Semester		Configured	Y	Y	
kuali.atp.type.Odd Years	Odd Years	Odd Numbered Years	Offering	Odd Years	Year		Configured	Y	Y	
kuali.atp.type.FallOddYears	Fall Odd Years	Fall Odd Years	Credit Course	Fall	Semester		Configured	Y	Y	

kuali.atp.type.SpringOddYears	Spring Odd Years	Spring Odd Years	Credit Course	Spring	Semester		Configured	Y	Y	
-------------------------------	------------------	------------------	---------------	--------	----------	--	------------	---	---	--

## (CM 2.0) Canonical Learning Unit (CLU) Service

- Class
- Description
- CLU Service Entity Diagram
- Service Contract Documentation
- Types and States
  - Course Types and States
  - Program Types and States
  - Standardized Test Types and State

### Class

Canonical Learning Unit Service is a **Class I** service. It provides the foundation for persistence for the:

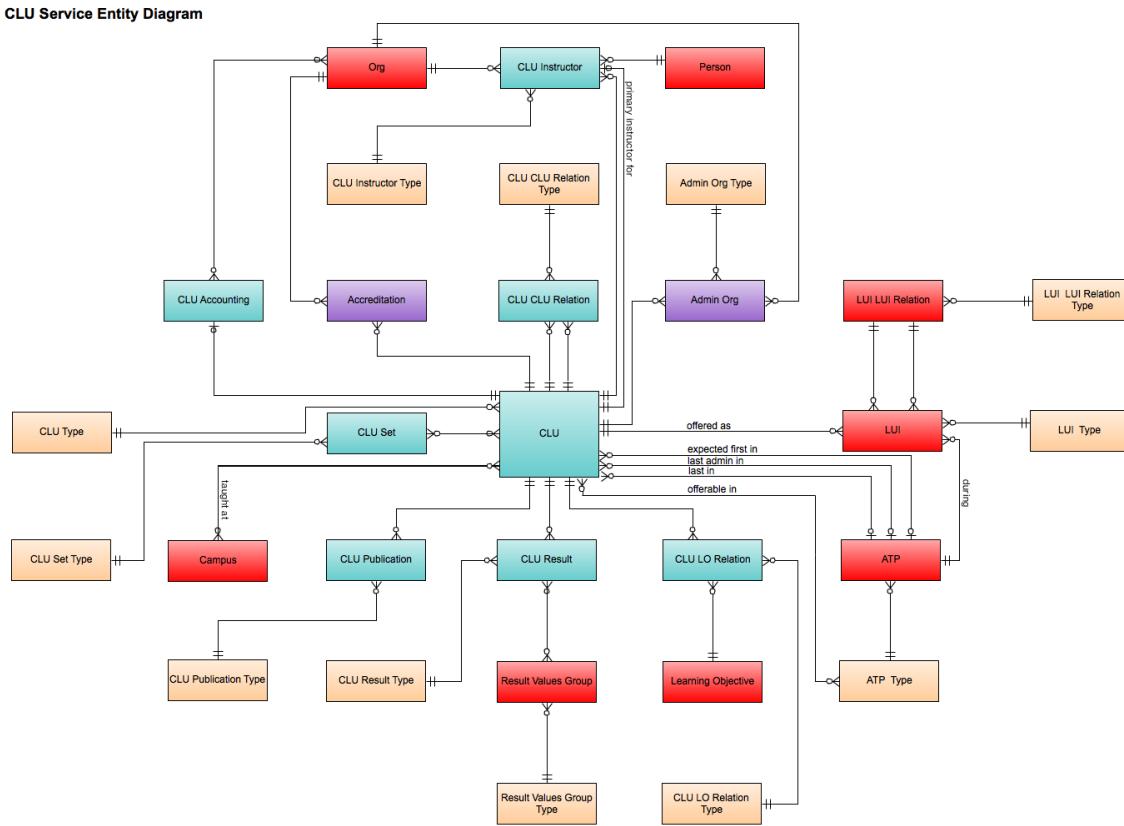
- (CM 2.0) Course Service
- (CM 2.0) Program Service

### Description

The Canonical Learning Unit Service supports the management of Canonical Learning Units (CLUs). This includes the development and approval of new Learning Units and substantive changes to existing Learning Units. A Learning Unit (LU) is any learning-related activity that needs to be tracked by the institution or the learner. Examples of learning units include the traditional curriculum of courses and degrees, professional and extension programs, and non-academic activities such as leadership development and service learning. All credit and non-credit learning activities, whether traditional or non-traditional, are learning units.

More...

### CLU Service Entity Diagram



More...

# **Service Contract Documentation**

[Go to Canonical Learning Unit \(CLU\) Service Contract Documentation](#)

## Types and States

## Course Types and States

More...

## Program Types and States

[More...](#)

## Standardized Test Types and State

More...

## **(CM 2.0) Canonical Learning Unit Service Descriptions and Assumptions**

## **Canonical Learning Unit (CLU) Service**

## Description

The Canonical Learning Unit Service supports the management of Canonical Learning Units (CLUs). This includes the development and approval of new Learning Units and substantive changes to existing Learning Units. A Learning Unit (LU) is any learning-related activity that needs to be

tracked by the institution or the learner. Examples of learning units include the traditional curriculum of courses and degrees, professional and extension programs, and non-academic activities such as leadership development and service learning. All credit and non-credit learning activities, whether traditional or non-traditional, are learning units.

Learning Units (LU) are first created at the canonical level as Canonical Learning Units (CLUs). When a CLU is offered during a Term (ATP), it is referred to as a Learning Unit Instance (LUI). LUIs are created and managed in the Learning Unit Instance Service. For example, an LU that represents a Course is created, proposed, and approved as a CLU and offered for a specific ATP as a LUI.

The CLU service is broad and covers many aspects of learning either directly or by reference (other services). The focus of this service initially is the creation of the inventory of CLUs that comprise an institution's offering including the associated learning objectives (outcome), learning results, supporting documents, evaluations, statements (structured rules logic specific to CLU), resource types and academic time periods.

## Key Concepts

### Canonical Learning Unit Types (CluType)

- Provide the basic structure or template for creating the CLU. Depending on the type, there may be different information associated with the CLU. Examples of CluTypes include Course, Program, Project, and Assessment. The granularity or CluType depends on the implementation.

### Canonical Learning Units (CLU)

- This is the inventory of Learning Units (LU) for an institution, the collection of LU that have been defined.
- These have generally been reviewed and approved through an approval process.
- There is a status (e.g., active, inactive, retired) associated with the "canonical" learning unit.
- CLUs are often approved for a range of Academic Time Periods.
- There are several ways to compose CLU into complex structures. LUI inherit the composition of the CLU.
  - CluClu Relations can be created to connect CLUs. The relation has a LuLuRelationType which is constrained by the CluType. This construct supports composite CLU, for example, a Chemistry class that is comprised of a Lecture and a Lab. CluClu Relations are used to define a series of CLU where all courses must be completed before credit is awarded.
  - CLU Sets are collections of CLU. They can be dynamic (based on a query) or a simple enumerated lists. CLU sets are unordered sets of CLU.

### Learning Unit Instance (LUI)

- The specific offering (occurrences) of a CLU typically during a specific Academic Time Period (ATP), though in some instances there may not be an associated calendar dates (e.g. online courses). These are managed in the Learning Unite Instance Service.

### Learning Objectives (LO)

- These objectives are the intended outcome(s) for the LU.
- LO are separate and related many-to-many to CLU, so a single CLU has multiple learning objectives and the same learning objective may be related to several CLUs.
  - There is a separate Learning Objective (LO) service that acts as a catalog service to manage the various LO that may be related to CLU.
  - LO may also be related directly to a Student, not part of this service (Current thinking is that this would be part of the Learning Plan (LP) which may be subsumed by the TBD Student Service).

### Result Options

- The wrapper for the result components defined within the Learning Result Catalog service.
- Each result option is associated with a specific result option usage type.
- CLUs and LUIs will not attach directly to result options.

### Result Sets

- The collection of one or more result options.
- The options in a set represent one permutation of results that can be offered by a CLU/LUI and achieved by a student.
- Each Result Set is associated with an LU Type to constrain which CLUs they can be attached to.

### CLU Results

- The association of CLUs to defined result set.
- The CLU Results represent the possible sets of learning results available for LUI instantiation.
- The CLU Results instantiated for a LUI in turn represent the available results for LPR.
- LU Statements can be evaluated during creation of an LPR in order to determine which results sets are valid for a specific person.

### Documents

- The associated supporting materials that accompany the CLU from inception, through approval, catalog listing, offering, evaluation and finally retiring.

- The CLU service manages the connection to documents but not the actual maintenance and storage of the documents themselves.
- They have been separated from CluInfo in support of different authorization access.

#### Requirement Components

- The basic building blocks for describing rules logic associated with various levels of LU (CluType, CLU and LUI).
- Each ReqComponent has a type that is the handshake with the BRMS which knows how to translate key/value pairs sent along with the type into the appropriate rules; for example, a type of LrdTime could represent a pre-req condition of having "taken CHEM1A within the last 6 months and received a grade of B or better."

#### Statements

- Reusable structured rule logic comprised of requirement components and/or other statements with a single logical operator, e.g. AND, OR to combine.
- Each statement is limited to one logical operator.
- The combination of nesting statement and/or requirement components supports the parentheses need for more complex logic.

#### Assumptions

- Type setup is part of configuration and not managed specifically through the service, including
  - CLuType - overall type of the CLU, e.g., course, academic course, exam, program, etc., specificity TBD as part of reference implementation
  - CLuCLuRelationType - relation type between CLUs and other CLUs, e.g. contains, precedes, etc.
  - CluResultType - description of the collection of result options contained within (currently based heavily off of the CluType)
  - ResultUsageType - expected usage type for the result option, e.g., final course grade, program certification
  - CluLoRelationType - relation type between CLUs and LOs (likely to be a single form for release 1)
  - CluSetType - type that can be used to distinguish between static (simple enumerated) and dynamic (query based) clu sets.
- Many of these types are constrained by other types; management of these constraints are also part of configuration and not managed through service operation, including these types which are constrained by CluType
  - CLuLuRelationType
  - CluResultType
  - CLuPublicationType
  - CluLoRelationType

#### Behavior and Interaction

- Retrieval operations do not have authorization checks. This needs to be confirmed with use cases to determine if this is the case.
- Relationships have a direction.
- Only learning objective identifiers are returned from the relevant CLU operations. Information on learning objectives are retrieved from the Learning Objective service.
- Only result component identifiers are referenced within the result options. Information on the result component and associated values are retrieved from the Learning Result Catalog service.

## (CM 2.0) Standardized Test Learning Unit Types and States

### Standardized Test Types

Type Key	Name	Description	Aliases	Agreed Upon	Notes/Comments
kuali.lu.type.standardized.test	Standardized Test	Standardized Test			
kuali.lu.type.standardized.test.component	Standardized Test Component	Standardized Test Component			

## (CM 2.0) Standardized Test Learning Unit Types

### Standardized Test Types

Type Key	Name	Description	Aliases	Agreed Upon	Notes/Comments
kuali.lu.type.standardized.test	Standardized Test	Standardized Test			
kuali.lu.type.standardized.test.component	Standardized Test Component	Standardized Test Component			

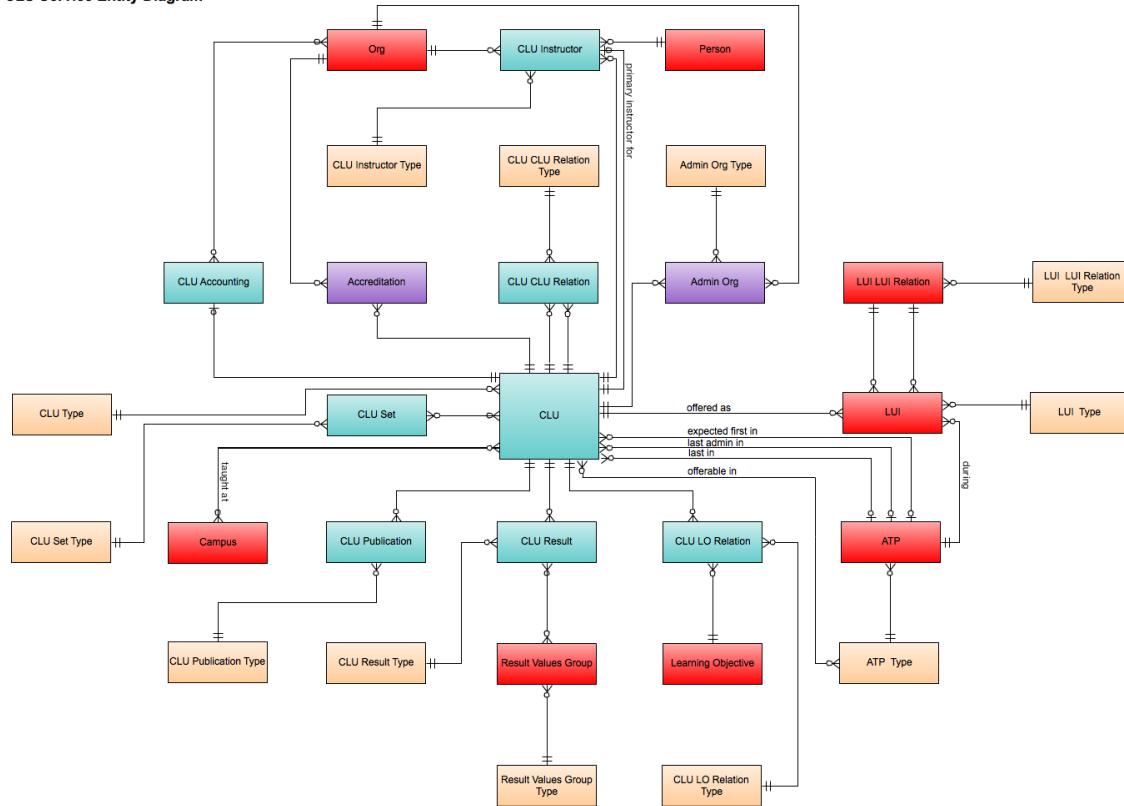
# (CM 2.0) Canonical Learning Unit Service Entity Diagram

- Entity Diagram
- Database Model and DDL

## Entity Diagram

Note: this image is out of date in terms of the LRC service and result options

CLU Service Entity Diagram



## Database Model and DDL

### (CM 2.0) Comment Service

- Class
- Description
- Entity Diagram
- Service Contract Documentation
- Type and State Configuration
  - Comment Types

## Class

Comment Service is a Core service that allows for the creation and management of user comments and tags associated with other objects across the system

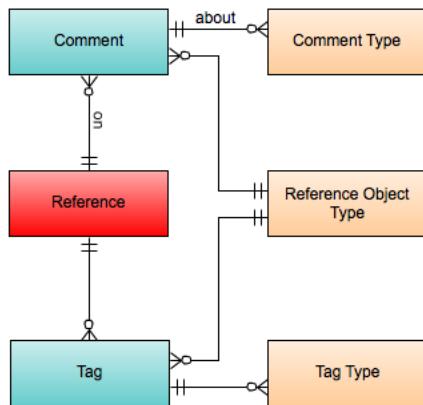
## Description

The Comment Service allows for the creation and management of user comments and tags associated with other objects across the system. There is no expectation that the objects know anything about the tags or comments; therefore objects can be deleted from the system even though tags or comments referencing those object may exist.

[More...](#)

## Entity Diagram

### Comment Service Entity Diagram



[More...](#)

## Service Contract Documentation

[Go to Comment Service Contract Documentation](#)

## Type and State Configuration

### Comment Types

Comment Type Key	Comment Type Name	Comment Type Description	Typical Usage	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.comment.type.Comment	Comment	Default			Placeholder	N	N	
kuali.comment.type.generalRemarks	General	General Comment			Placeholder	N	N	
kuali.comment.type.workflowDecisionRationale.acknowledge	Ack Decision	Decision Rationale Ack			Placeholder	N	N	
kuali.comment.type.workflowDecisionRationale.approve	Approve Decision	Decision Rationale Approve			Placeholder	N	N	

kuali.commont.type.workflowDecision.Rationale.blanketApprove	Blanket Approve	Decision Rationale Blanket Approve Proposal			Placeholder	N	N	
kuali.commont.type.workflowDecision.Rationale.cancelWorkflow	Cancel Proposal	Decision Rationale Cancel Proposal			Placeholder	N	N	
kuali.commont.type.workflowDecision.Rationale.fyi	FYI Decision	Decision Rationale FYI			Placeholder	N	N	
kuali.commont.type.workflowDecision.Rationale.reject	Reject Decision	Decision Rationale Reject			Placeholder	N	N	
kuali.commont.type.workflowDecision.Rationale.return	Return Decision	Decision Rationale Return			Placeholder	N	N	
kuali.commont.type.workflowDecision.Rationale.withdraw	Withdraw Decision	Decision Rationale Withdraw			Placeholder	N	N	

More...

## (CM 2.0) Comment Service Description and Assumptions

### Comment Service

#### Description

The Comment Service allows for the creation and management of user comments and tags associated with other objects across the system. There is no expectation that the objects know anything about the tags or comments; therefore objects can be deleted from the system even though tags or comments referencing those object may exist.

#### Key Concepts

##### Comment

- A comment is a block of text that can be associated with certain objects often as part of a collaborative or workflow process.
- The comment type can be used to help the application differentiate comments that might need to be displayed in particular contexts, such as during Course Approval.

##### Tag

- A kind of metadata which helps categorize or describe an item and may allow it to be found again by browsing or searching.
- Each tag is contains of a multipart key consisting of a namespace, a predicate and a value.
- User profile-specific tags can be accomplished with the existing tag construct, e.g., namespace for a principal, etc.
- The tag type will likely only be a single value for tag type.
  - The justification for adding it is to maintain consistency in being able to describe the structure with the dictionary operations.

##### Reference

- A pointer to an object that exists elsewhere in the system, e.g., aCLU, a document, etc.
- The reference type indicates the type of object that is referenced by the comment or tag and is part of a composite id/type key for the reference.

## Assumptions

- Type setup is part of configuration and not managed specifically through the service, including commentType, tagType and referenceType.
- Comments and tags are linked to other objects in the system by referencing the object's id and type.
- Comments are always associated with a single reference id and reference type.
- Comments are not nested/threaded.
- Tags may be associated with several different reference ids and reference types indirectly as several tags may share the same tag information.

## (CM 2.0) Comment Service Types and States

### Types

#### Comment Types

Comment Type Key	Comment Type Name	Comment Type Description	Typical Usage	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.comment.type.Comment	Comment	Default			Placeholder	N	N	
kuali.comment.type.generalRemarks	General	General Comment			Placeholder	N	N	
kuali.comment.type.workflowDecision.Rationale.acknowledge	Ack Decision	Decision Rationale Ack			Placeholder	N	N	
kuali.comment.type.workflowDecision.Rationale.approve	Approve Decision	Decision Rationale Approve			Placeholder	N	N	
kuali.comment.type.workflowDecision.Rationale.blanketApprove	Blanket Approve	Decision Rationale Blanket Approve Proposal			Placeholder	N	N	
kuali.comment.type.workflowDecision.Rationale.cancelWorkflow	Cancel Proposal	Decision Rationale Cancel Proposal			Placeholder	N	N	
kuali.comment.type.workflowDecision.Rationale.fyi	FYI Decision	Decision Rationale FYI			Placeholder	N	N	
kuali.comment.type.workflowDecision.Rationale.reject	Reject Decision	Decision Rationale Reject			Placeholder	N	N	
kuali.comment.type.workflowDecision.Rationale.return	Return Decision	Decision Rationale Return			Placeholder	N	N	

kuali.comment.type.workflowDecision.Rationale.withdraw	Withdraw Decision	Decision Rationale Withdraw			Placeholder	N	N	
--------------------------------------------------------	-------------------	-----------------------------	--	--	-------------	---	---	--

## States

### Comment States

Process Key	
kuali.comment.process	

State Key	State Name	State Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.comment.state.active (initial)	Active	Indicates that this comment is active		Placeholder	N	N	
kuali.comment.state.inactive	Inactive	Indicates that this comment is inactive	Removed	Placeholder	N	N	

## (CM 2.0) Comment States

### Comment States

Process Key	
kuali.comment.process	

State Key	State Name	State Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.comment.state.active (initial)	Active	Indicates that this comment is active		Placeholder	N	N	
kuali.comment.state.inactive	Inactive	Indicates that this comment is inactive	Removed	Placeholder	N	N	

## (CM 2.0) Comment Types

### Comment Types

Comment Type Key	Comment Type Name	Comment Type Description	Typical Usage	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.comment.type.Comment	Comment	Default			Placeholder	N	N	

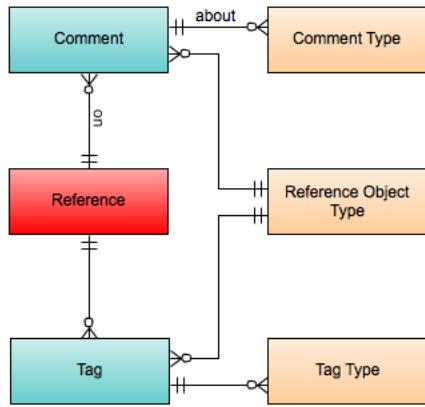
kuali.comment.type.generalRemarks	General	General Comment			Placeholder	N	N	
kuali.comment.type.workflowDecision.Rationale.acknowledge	Ack Decision	Decision Rationale Ack			Placeholder	N	N	
kuali.comment.type.workflowDecision.Rationale.approve	Approve Decision	Decision Rationale Approve			Placeholder	N	N	
kuali.comment.type.workflowDecision.Rationale.blanketApprove	Blanket Approve	Decision Rationale Blanket Approve Proposal			Placeholder	N	N	
kuali.comment.type.workflowDecision.Rationale.cancelWorkflow	Cancel Proposal	Decision Rationale Cancel Proposal			Placeholder	N	N	
kuali.comment.type.workflowDecision.Rationale.fyi	FYI Decision	Decision Rationale FYI			Placeholder	N	N	
kuali.comment.type.workflowDecision.Rationale.reject	Reject Decision	Decision Rationale Reject			Placeholder	N	N	
kuali.comment.type.workflowDecision.Rationale.return	Return Decision	Decision Rationale Return			Placeholder	N	N	
kuali.comment.type.workflowDecision.Rationale.withdraw	Withdraw Decision	Decision Rationale Withdraw			Placeholder	N	N	

## (CM 2.0) Comment Service Entity Diagram

- Entity Diagram
- DDL and Entity Diagram for Enrollment

### Entity Diagram

## Comment Service Entity Diagram



## DDL and Entity Diagram for Enrollment

KSEN\_COMMENT.pdf  
KSEN\_COMMENT\_DDL\_Tables.ddl  
KSEN\_COMMENT\_DDL\_Constraints.ddl

## (CM 2.0) Course Service

- Class
- Description
- Entity Diagram
- Layering Diagram
- Service Contract Documentation
- Type and State Configuration

### Class

Course Service is a Class II service that provides a more business oriented view of the data managed in the Learning Unit (LU) Service.

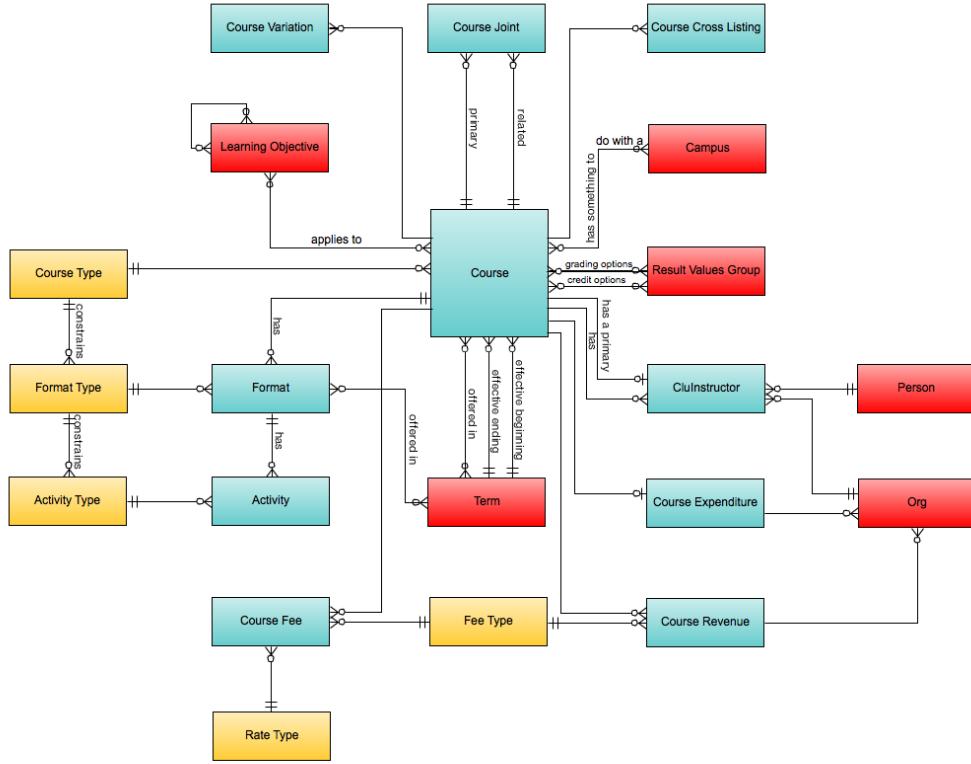
### Description

A course is a specified area of knowledge contained and taught independently from other areas of knowledge that may or may not be related. Courses typically have specific learning objectives, defined student learning outcomes and assessments. They can be recognized for academic credit or not. Courses can be delivered in various instructional formats such as lecture, lab, discussion, seminar, colloquium, etc. A course would utilize at least one instructional format, but often will use more than one.

[More...](#)

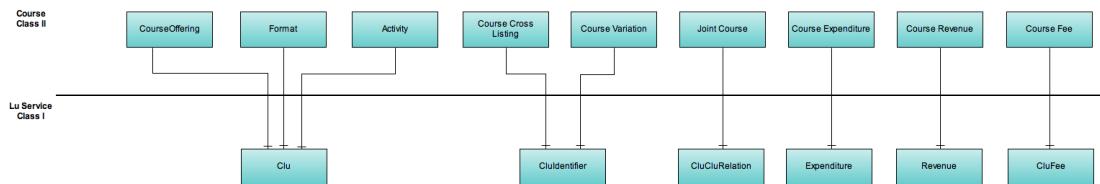
### Entity Diagram

## Course Service Entity Diagram



[More...](#)

## Layering Diagram



[More...](#)

## Service Contract Documentation

[Go to Course Service Contract Documentation](#)

## Type and State Configuration

[More...](#)

## (CM 2.0) Course Service Description and Assumptions

## Course Service

### Description

A course is a specified area of knowledge contained and taught independently from other areas of knowledge that may or may not be related. Courses typically have specific learning objectives, defined student learning outcomes and assessments. They can be recognized for academic credit or not. Courses can be delivered in various instructional formats such as lecture, lab, discussion, seminar, colloquium, etc. A course would utilize at least one instructional format, but often will use more than one.

Course is pre-configured type of CLU (Canonical Learning Unit) that represents a credit course. The courseInfo object supports a number of complex course options – cross-listed, joint and variations. Course can have any number of associated learning objectives and requisites (rules that model pre-, co-, anti-requisites and enrollment restrictions).

### Key Concepts

#### Course, Format and Activity

- Each course has one or more formats, each of which can have one or more activities.
- The activity represents the specific teaching experience through which learning is imparted to the student (i.e. Lecture, Lab, etc).
- The format describes the "footprint" of activities that describe the course.

#### Financials

- Financial information is associated with the course and not the format or activity. It includes the fee justification, list of fees, associated revenues and expenditure (org) distributions.

#### Grading and Credits

- Grading and credits are associated with the course and not the format or activity.
- Multiple grading (e.g., letter, pass/not pass) and credit (fixed/variable with units) options can be specified.

#### Complex Courses

- Any number of joint courses (code, ID and description) can be specified for a course; these courses must already exist.
- Cross-listed courses are modeled as alternate IDs related to the course entity; multiple cross-listed IDs can be specified.
- Variations are modeled as additional "suffixes" including the variation code and the related title.

#### Learning Objectives

- Multiple Learning Objectives can be associated with courses. These can be structured, from a managed catalog, or added (and reused) ad hoc as part of defining each course.
- The actual management of the LO is managed in a separate Learning Objective service. The course service provides the mechanism for associating LOs to the course.

#### Course Requisites

- Multiple Course requisites, e.g., Pre-reqs and enrollment restrictions, can be associated with courses.
- The actual management of the requisites is managed in a separate Statement service. The course service provides the mechanism for associating requisites to the course.

#### Assumptions

- The focus for CM is "credit courses" which is a type of CLU.
- Type setup is part of configuration and not managed specifically through the service, including courseType, formatType, activityType, feeType, rateType.

#### Deferred Design

- Non standard courses
  - Spans multiple terms
  - Research Project
  - Thesis

## (CM 2.0) Course Types and States

### Course Types

Type Key	Name	Description	Aliases	Agreed Upon	In Constant File?	In Database?	Notes/Comments
kuali.lu.type.CreditCourse	Credit Course	An course offered for academic credit		Configured	Y	Y	
kuali.lu.type.NonCreditCourse	Non-Credit Course	A course that is not offered for regular academic credit		Placeholder	N	N	
<b>Course Format Types</b>							
kuali.lu.type.CreditCourseFormatShell	Credit Course Format Shell	A shell for course formats		Configured	Y	Y	
<b>Course Activity Types</b>							
kuali.lu.type.activity.Clerkship	Clerkship			Configured - Core Slice	N	Y	 <b>KSCM-1322</b> - CM: review/compare all shared Types between CM and ENR for discrepancies/change  <a href="#">Open</a>
kuali.lu.type.activity.Clinic	Clinic			Configured - Core Slice	N	Y	 <b>KSCM-1322</b> - CM: review/compare all shared Types between CM and ENR for discrepancies/change  <a href="#">Open</a>
kuali.lu.type.activity.Conference	Conference			Configured - Core Slice	N	Y	 <b>KSCM-1322</b> - CM: review/compare all shared Types between CM and ENR for discrepancies/change  <a href="#">Open</a>
kuali.lu.type.activity.Directed	Directed	The exchange of opinions or questions on course material, directed by the instructor.		Configured	Y	Y	

kuali.lu.type.activity.Discussion	Discussion	The exchange of opinions or questions on course material.		Configured	Y	Y	
kuali.lu.type.activity.Homework	Homework	Student's doing homework, problem sets and reading assignments and writing		Configured - Core Slice	N	Y	
kuali.lu.type.activity.IndependentStudy	Independent Study			Configured - Core Slice	N	Y	
kuali.lu.type.activity.Lab	Lab	Student working on projects in a defined laboratory space. Instructors are on-hand for students to ask questions and guidance.		Configured	Y	Y	
kuali.lu.type.activity.Lecture	Lecture	Instructor presentation of course materials.		Configured	Y	Y	
kuali.lu.type.activity.Practicum	Practicum			Configured - Core Slice	N	Y	 <b>KSCM-1322</b> - CM: review/compare all shared Types between CM and ENR for discrepancies/change  <a href="#">Open</a>
kuali.lu.type.activity.QuiZ	Quiz			Configured - Core Slice	N	Y	 <b>KSCM-1322</b> - CM: review/compare all shared Types between CM and ENR for discrepancies/change  <a href="#">Open</a>
kuali.lu.type.activity.Seminar	Seminar			Configured - Core Slice	N	Y	 <b>KSCM-1322</b> - CM: review/compare all shared Types between CM and ENR for discrepancies/change  <a href="#">Open</a>

kuali.lu.type.activity.Studio	Studio			Configured - Core Slice	N	Y	 <b>KSCM-1322</b> - CM: review/compare all shared Types between CM and ENR for discrepancies/change <a href="#">(Open)</a>
kuali.lu.type.activity.Tutorial	Tutorial	Instructor or assistant walking through a learning practice.		Configured	Y	Y	
kuali.lu.type.activity.Web Discussion	Web Discuss	Web-based or technologically-mediated activities replacing standard discussion sections		Configured	Y	Y	
kuali.lu.type.activity.Web Lecture	Web Lecture	Instructor presentation of course materials via the web		Configured	Y	Y	

## Types used in testing in during CM but somehow got in released code

IuType.shell.course	Course	A Shell Course
IuType.shell.program	Program	A Shell Program

## (CM 2.0) Course Types

### Course Types

Type Key	Name	Description	Aliases	Agreed Upon	In Constant File?	In Database?	Notes/Comments
kuali.lu.type.CreditCourse	Credit Course	An course offered for academic credit		Configured	Y	Y	
kuali.lu.type.NonCreditCourse	Non-Credit Course	A course that is not offered for regular academic credit		Placeholder	N	N	
<b>Course Format Types</b>							
kuali.lu.type.CreditCourseFormatShell	Credit Course Format Shell	A shell for course formats		Configured	Y	Y	

Course Activity Types							
kuali.lu.type.activity.Clerkship	Clerkship			Configured - Core Slice	N	Y	 <b>KSCM-1322</b> - CM: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a> )
kuali.lu.type.activity.Clinic	Clinic			Configured - Core Slice	N	Y	 <b>KSCM-1322</b> - CM: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a> )
kuali.lu.type.activity.Conference	Conference			Configured - Core Slice	N	Y	 <b>KSCM-1322</b> - CM: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a> )
kuali.lu.type.activity.Directed	Directed	The exchange of opinions or questions on course material, directed by the instructor.		Configured	Y	Y	
kuali.lu.type.activity.Discussion	Discussion	The exchange of opinions or questions on course material.		Configured	Y	Y	
kuali.lu.type.activity.Homework	Homework	Student's doing homework, problem sets and reading assignments and writing		Configured - Core Slice	N	Y	
kuali.lu.type.activity.IndependentStudy	Independent Study			Configured - Core Slice	N	Y	

kuali.lu.type.activity.Lab	Lab	Student working on projects in a defined laboratory space. Instructors are on-hand for students to ask questions and guidance.		Configured	Y	Y	
kuali.lu.type.activity.Lecture	Lecture	Instructor presentation of course materials.		Configured	Y	Y	
kuali.lu.type.activity.Practicum	Practicum			Configured - Core Slice	N	Y	 <b>KSCM-1322</b> - CM: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a> )
kuali.lu.type.activity.Quiz	Quiz			Configured - Core Slice	N	Y	 <b>KSCM-1322</b> - CM: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a> )
kuali.lu.type.activity.Seminar	Seminar			Configured - Core Slice	N	Y	 <b>KSCM-1322</b> - CM: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a> )
kuali.lu.type.activity.Studio	Studio			Configured - Core Slice	N	Y	 <b>KSCM-1322</b> - CM: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a> )
kuali.lu.type.activity.Tutorial	Tutorial	Instructor or assistant walking through a learning practice.		Configured	Y	Y	

kuali.lu.type.activity.Web Discussion	Web Discuss	Web-based or technologically-mediated activities replacing standard discussion sections		Configured	Y	Y	
kuali.lu.type.activity.Web Lecture	Web Lecture	Instructor presentation of course materials via the web		Configured	Y	Y	

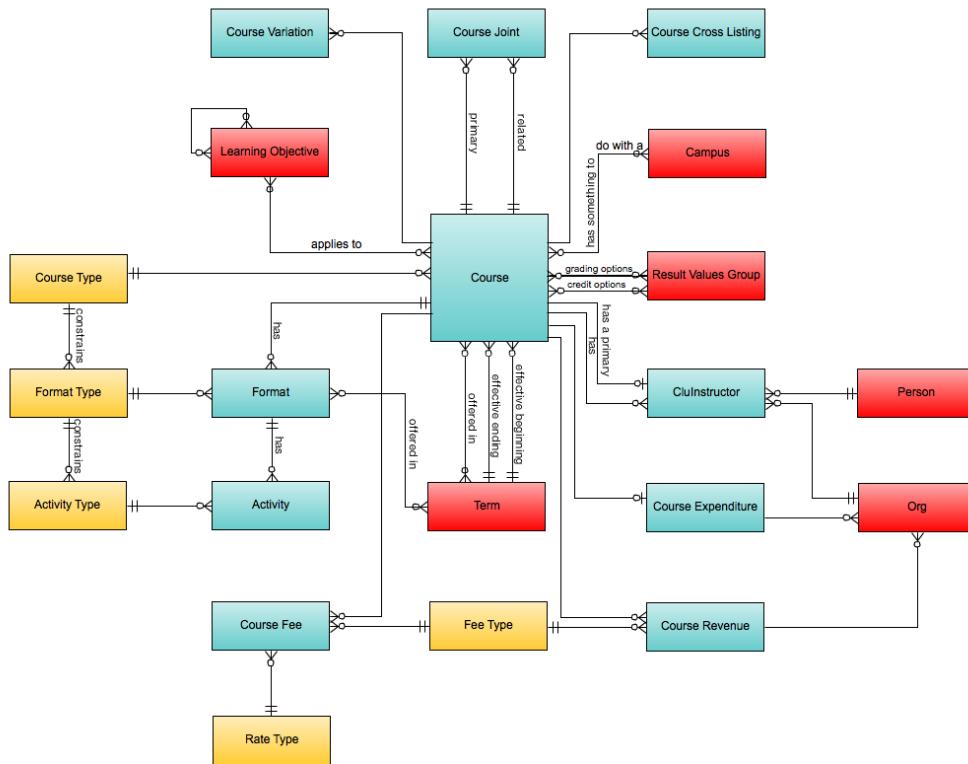
Types used in testing in during CM but somehow got in released code

luType.shell.course	Course	A Shell Course
luType.shell.program	Program	A Shell Program

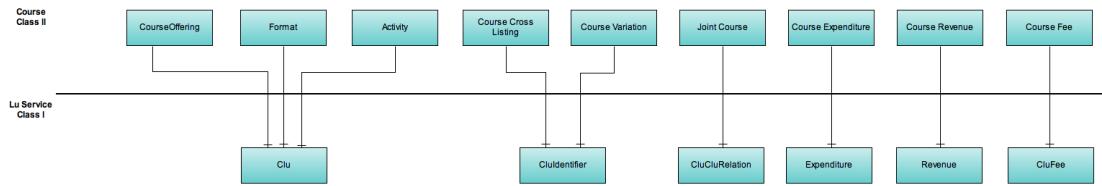
## (CM 2.0) Course Service Entity Diagram

### Entity Diagram

Course Service Entity Diagram



### Layering Diagram



## (CM 2.0) Document Service

- Class
- Description
- Entity Diagram
- Service Contract Documentation
- Type and State Configuration

### Class

Document Service is a Core service that supports the management of document objects.

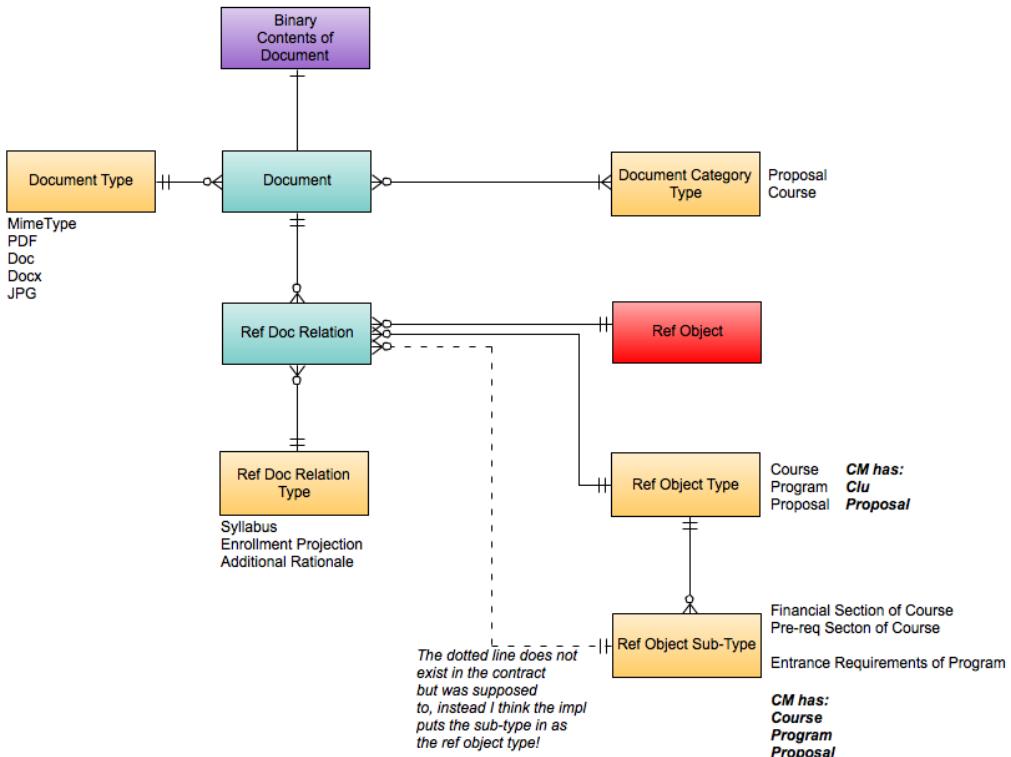
### Description

The page (CM 2.0) (CM 2.0) Document Service Description and Assumptions could not be found.

[More...](#)

### Entity Diagram

#### Document Service Entity Diagram



### Service Contract Documentation

## Type and State Configuration

The page (CM 2.0) (CM 2.0) Document Service Types and States could not be found.

[More...](#)

# (CM 2.0) Document Service Description and Assumptions

## Document Service

### Description

The Document Service supports the management of document objects. Relations between stored documents and external entities are managed through the respective entity service.

### Key Concepts

#### Document Type

- The document type indicates the format of the document, e.g., text, jpeg, xml
- The usage of types within this service is similar to MIME types

#### Document Category

- Categories are used in this service for classification and organization of documents
- Upon creation, the document must be associated with a category
- Additional categories can be added or removed as long as each document is associated with at least one category

#### Retrieving Document Information

- The "Read" operations with the service are used to retrieve document(s) when the document identifier is known
- Output data includes both the meta-information and the document binary
- In cases where the service caller is interested in retrieving custom field sets using criteria other than the document identifier, custom search results and search types can be configured for the searchForResults operation

### Assumptions

- Documents always have at least one category and are not orphaned
- Type setup is part of configuration and not managed specifically through the service, including documentType, documentCategoryType, refDocRelationType, refObjectType and refObjectSubTYpe

# (CM 2.0) Document Service Types and States

- Document Categories
- Document Types
- Ref Object Types
- Ref Object Sub-Types
- Ref Doc Relation Types

## Document Categories

Category Key	Name	Description	Status	Notes/Questions
documentCategory.course	Course Documents	Documents attached to a proposal	Configured in CM	<p>The keys do not conform to the new R2 standard of starting with kuali.document.category</p> <p> <b>KSCM-1322 - C</b> M: review/compare all shared Types between CM and ENR for discrepancies/change (  <b>Open</b>)</p>

documentCategory.proposal	Proposal Documents	Documents attached to a Course	Configured in CM	<input checked="" type="checkbox"/> <b>KSCM-1322 - C</b> M: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a> )
---------------------------	--------------------	--------------------------------	------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Document Types

Key	Name	Description	Status	Notes/Questions
documentType.doc	doc	doc Document	Configured in CM	<input checked="" type="checkbox"/> <b>KSCM-1322 - C</b> M: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a> )
documentType.docx	docx	docx Document	Configured in CM	<input checked="" type="checkbox"/> <b>KSCM-1322 - C</b> M: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a> )
documentType.gif	gif	gif Document	Configured in CM	<input checked="" type="checkbox"/> <b>KSCM-1322 - C</b> M: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a> )
documentType.gz	gz	gz Document	Configured in CM	<input checked="" type="checkbox"/> <b>KSCM-1322 - C</b> M: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a> )
documentType.jpg	jpg	jpg Document	Configured in CM	<input checked="" type="checkbox"/> <b>KSCM-1322 - C</b> M: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a> )
documentType.mp3	mp3	mp3 Document	Configured in CM	<input checked="" type="checkbox"/> <b>KSCM-1322 - C</b> M: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a> )

documentType.pdf	pdf	pdf Document	Configured in CM	<b>KSCM-1322 - C</b> M: review/compare all shared Types between CM and ENR for discrepancies/change ( <a href="#">Open</a> )
documentType.png	png	png Document	Configured in CM	<b>KSCM-1322 - C</b> M: review/compare all shared Types between CM and ENR for discrepancies/change ( <a href="#">Open</a> )
documentType.ppt	ppt	ppt Document	Configured in CM	
documentType.rar	rar	rar Document	Configured in CM	
documentType.rtf	rtf	rtf Document	Configured in CM	
documentType.tar	tar	tar Document	Configured in CM	
documentType.tif	tif	tif Document	Configured in CM	
documentType.tiff	tiff	tiff Document	Configured in CM	
documentType.txt	txt	txt Document	Configured in CM	
documentType.wav	wav	wav Document	Configured in CM	
documentType.xls	xls	xls Document	Configured in CM	
documentType.xml	xml	xml Document	Configured in CM	
documentType.zip	zip	zip Document	Configured in CM	

## Ref Object Types

Key	Name	Description	Status	Notes
kuali.org.RefObjectType.CluInfo	CluInfo	Clu Object Type	Configured in CM	These do not conform to the new R2 standard of kuali.ref.object.type. <b>KSCM-1322 - C</b> M: review/compare all shared Types between CM and ENR for discrepancies/change ( <a href="#">Open</a> )
kuali.org.RefObjectType.ProposalInfo	ProposalInfo	Proposal Object Type	Configured in CM	<b>KSCM-1322 - C</b> M: review/compare all shared Types between CM and ENR for discrepancies/change ( <a href="#">Open</a> )

## Ref Object Sub-Types

Key	Name	Description	Status	Notes/Comments
-----	------	-------------	--------	----------------

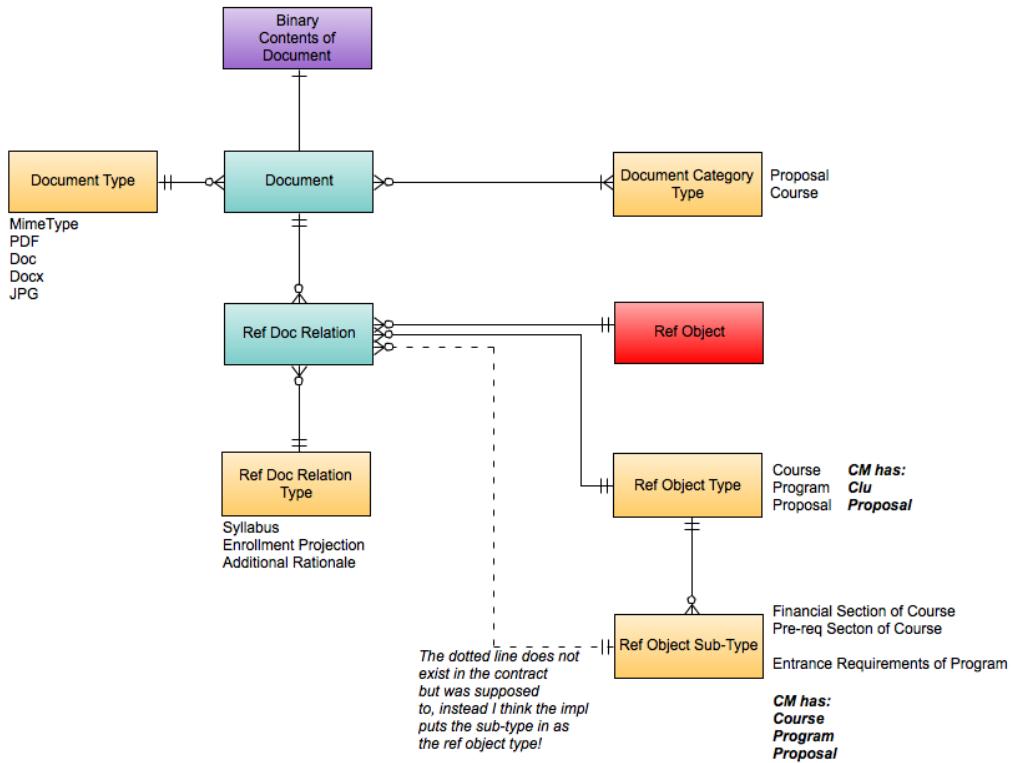
kuali.org.RefObjectSubType.Course	Course	Sub Type for Course	Configured in CM	Does not conform the new R2 standard kuali.ref.object.sub.type.  <span style="border: 1px solid #ccc; padding: 2px;">☒ <b>KSCM-1322 - C</b> M: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a>)</span>
kuali.org.RefObjectSubType.Program	Program	Sub Type for Program	Configured in CM	These types are supposed to point to parts of a course or program or proposal, for example the financial section  <span style="border: 1px solid #ccc; padding: 2px;">☒ <b>KSCM-1322 - C</b> M: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a>)</span>
kuali.org.RefObjectSubType.Proposal	Proposal	Sub Type for Proposal	Configured in CM	  <span style="border: 1px solid #ccc; padding: 2px;">☒ <b>KSCM-1322 - C</b> M: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a>)</span>

## Ref Doc Relation Types

Key	Name	Description	Status	Notes
kuali.org.DocRelation.allObjectTypes	All Relations	Relation for all known sub object types	Configured in CM	These do not conform to the new R2 standard of kuali.ref.doc.relation.type.  <span style="border: 1px solid #ccc; padding: 2px;">☒ <b>KSCM-1322 - C</b> M: review/compare all shared Types between CM and ENR for discrepancies/change (  <a href="#">Open</a>)</span>

## (CM 2.0) Document Service Entity Diagram

## Document Service Entity Diagram



## (CM 2.0) Enumeration Management Service

- Class
- Description
- Entity Diagram
- Service Contract Documentation
- Type and State Configuration

### Class

Enumeration Management Service is a Core service that supports the management of code tables for other services.

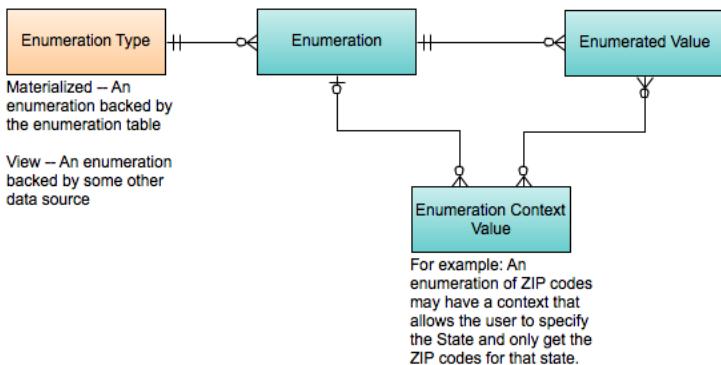
### Description

Enumeration Management service is only accessed by authorized callers configuring some piece of the system. The *code* identifies the value within an enumeration.

[More...](#)

### Entity Diagram

## Enumeration Management Service Entity Diagram



[More...](#)

## Service Contract Documentation

[Go to Enumeration Management Service Contract Documentation](#)

## Type and State Configuration

TBD

[More...](#)

## (CM 2.0) Enumeration Management Service Description and Assumptions

### Enumeration Management Service

#### Description

Enumeration Management service is only accessed by authorized callers configuring some piece of the system. The *code* identifies the value within an enumeration.

#### Key Concepts

- This service only supports retrieval of enumerations which are managed through this service.
- Only manually maintained enumerations are supported by this service; enumerations which act as views of configuration information or views of existing objects are maintained by the service of record.

#### Assumptions

- The creation/maintenance of an enumeration, not the code/value pairs that comprise it, is handled via configuration to minimize secondary issues.
- Creation of an enumeration typically does not provide utility outside of secondary changes which involve configuration tasksenumerationType.
  - For example, if an enumeration is intended to constrain the list of values available for a particular field in a service, the dictionary for the type containing that field must be updated to reference the enumeration.
  - Alteration of the constraints on the fields in the enumeration may lead to ripple effects, forcing secondary updates through all the services referencing the enumeration.
- While a delete operation is provided, values in general should not be deleted but have their expiration date set to inactive.
- Type setup is part of configuration and not managed specifically through the service, including enumerationType.

## (CM 2.0) Enumeration Management Service Types and States

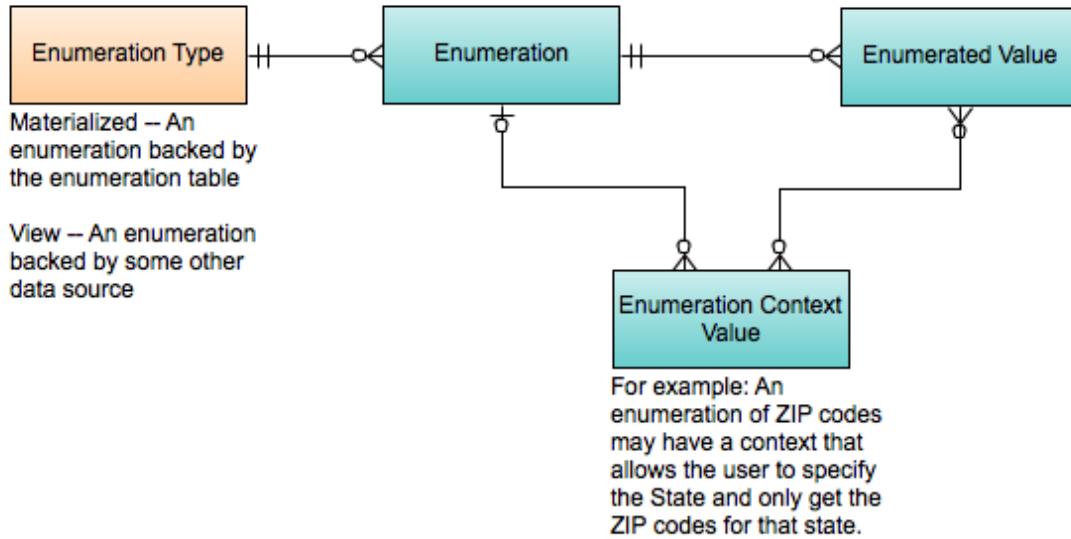
TBD

## (CM 2.0) Enumeration Management Entity Model

- Enumeration Management Entity Model
- Database Model and DDL Scripts

### Enumeration Management Entity Model

#### Enumeration Management Service Entity Diagram



### Database Model and DDL Scripts

- KSEN\_ENUMERATION.pdf
- KSEN\_ENUMERATION\_DDL\_Tables.ddl
- KSEN\_ENUMERATION\_DDL\_Constraints.ddl

## (CM 2.0) Learning Objective (LO) Service

- Class
- Description
- Entity Diagram
- Service Contract Documentation
- Type and State Configuration
  - LO Types
  - LO States

### Class

Learning Objective is a Class I service supports the management of learning objectives.

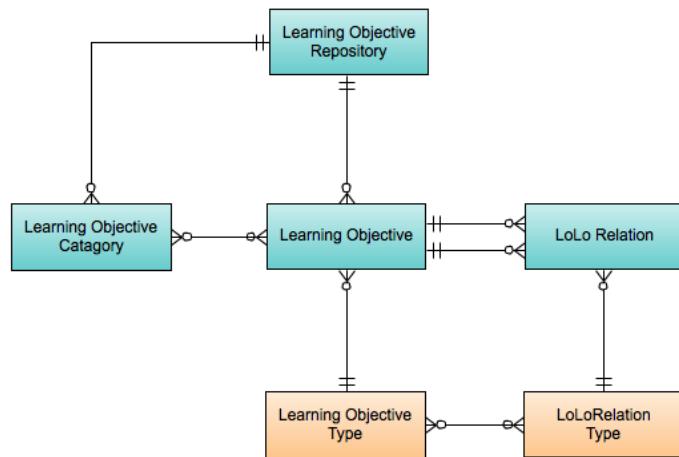
## Description

The Learning Objective Service supports the management of formal learning objectives. This is primarily a catalog service with basic operations.

[More...](#)

## Entity Diagram

**Learning Objective Service Entity Diagram**



[More...](#)

## Service Contract Documentation

[Go to Learning Objective Service Contract Documentation](#)

## Type and State Configuration

@Norm – can you help with some type state?

### LO Types

### LO States

[More...](#)

## (CM 2.0) Learning Objective Service Description and Assumptions

### Learning Objective Service

#### Description

The Learning Objective Service supports the management of formal learning objectives. This is primarily a catalog service with basic operations.

#### Key Concepts

##### Learning Objectives in a Repository

- A Repository exists to provide a way to enable different policy and authorization rules to be established. For example, you might have one repository for externally maintained LOs, and another for your institutions. The external ones probably are only maintained by a couple of individuals as the external source make changes, but you can have an entirely different set of rules around your local LOs.

- It is not mandatory that the LOs are all arranged in a tree, it is possible to have a flat repository which has no root.
- A learning objective is attached to only one repository, and can optionally be at the root or as a relation with another learning objective. For an existing learning objective, it can be moved to another node or attached to multiple LOs within the same repository.

#### **LO LO Relations**

- A relation between LOs and other LOs e.g., supports/in support of, is equivalent to, etc.
- This is a directional reference that can be used between learning objectives in the same repository or different repositories.
- For a bi-directional equivalency, the equivalency reference must be individually established on the learning objective relation type (using both name and revName fields on the loLoRelationTypeInfo structure).

#### **Learning Objective Categories**

- Categories can be created and attached to a single learning objective repository as a means for grouping learning objectives within the same repository.
- Learning objectives can be associated with one or more categories within the same repository.

#### **Assumptions**

- Read operations will not be restricted – no authorizations needed.
- Authorizations for maintaining learning objectives will be assigned by the repository.
- Creating repositories, including the "root" node for the repository, is handled in configuration.
- Operations to link learning objectives to outside entities will be handled in the respective service, e.g. linking a CLU to a learning objective is handled in the LU Service.
- Type setup is part of configuration and not managed specifically through the service, including learningObjectType and loLoRelationType.

## **(CM 2.0) Learning Objective Types and States**

This page holds the agreed upon Type and State configuration for Learning Objectives

- [LO Types](#)
- [LO States](#)

@Norm – can you help with some type state?

#### **LO Types**

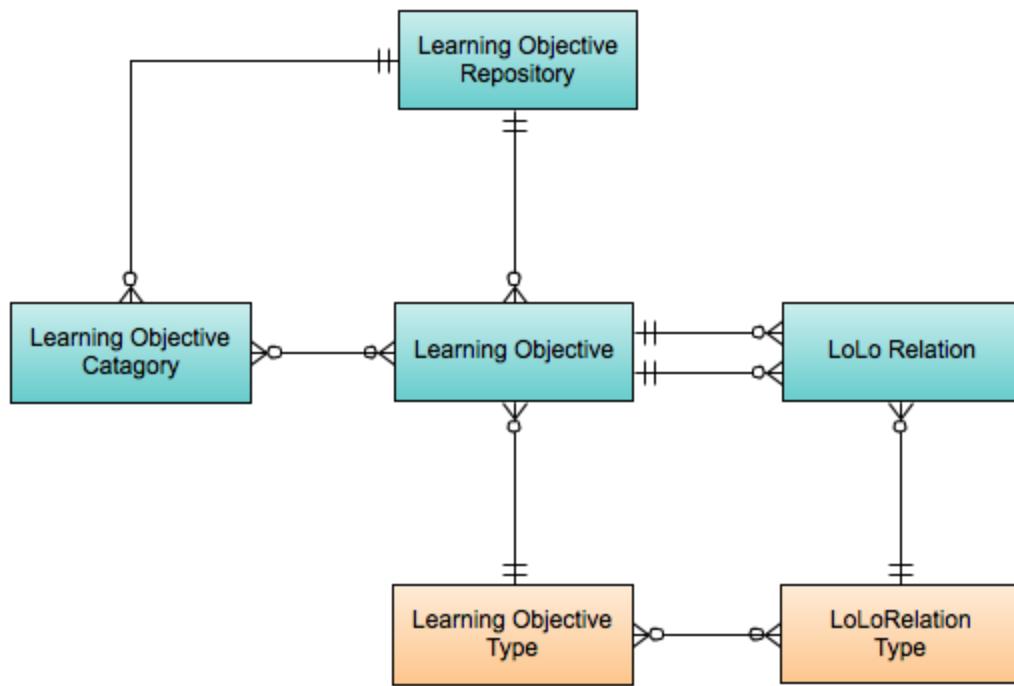
#### **LO States**

## **(CM 2.0) Learning Objective Service Entity Diagram**

- [Learning Objective Service Entity Diagram](#)
- [Data Model](#)

#### **Learning Objective Service Entity Diagram**

## Learning Objective Service Entity Diagram



### Data Model

- KSEN\_LO.pdf
- KSEN\_LO\_DDL\_Tables.ddl
- KSEN\_LO\_DDL\_Constraints.ddl

## (CM 2.0) Learning Result Catalog (LRC) Service

- Class
- Description
- Entity Diagram
- Service Contract Documentation
- Types and States
- Change Log
- Developer Documentation

### Class

Learning Result Catalog Service is a Class I service

### Description

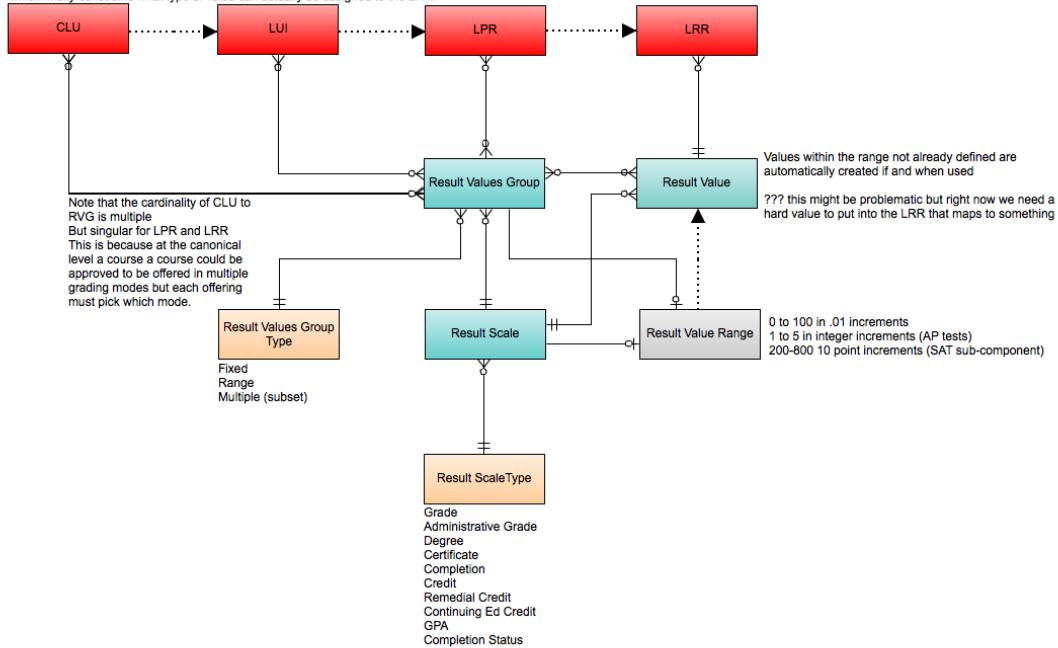
The Learning Result Catalog service supports the management of learning results. It provides the basis for defining the result types and values that can be associated with the catalog LU (CLU), the offered LU (LUI), and the specific student (LPR).

[More...](#)

# Entity Diagram

## Learning Result Catalog Service Entity Diagram

The Result Values Groups (RVGs) defined in CLU constrains the allowed RVG in LUI which constrains the RVG in LPR which constrains the RVG in LRR which finally constrains what type of value can actually be assigned to the LRR.



[More...](#)

## Service Contract Documentation

[Go to Learning Result Catalog Service Contract Documentation](#)

## Types and States

[More...](#)

## Change Log

The LRC was significantly redesigned from R1 to R2 with the following goals in mind:

1. Simplification – the old LRC mixed class I concepts such as Result Component with Class II concepts such as Grade and Credits leading to confusion by developers
2. Better ability to logically group values for different purposes – mix and match them
3. Better management of ranges of values by being able to specify the range and increment instead of having to specify all possible values within the range
4. The types were straightened out so they don't mix up multiple concepts

[More...](#)

## Developer Documentation

The page (CM 2.0) Learning Result Catalog Developer Documentation could not be found.

[More...](#)

## (CM 2.0) Learning Result Catalog Change Log

- Summary of Changes from R1 to R2

- Goals
- Objects
- Types
- Methods
- Global Changes to all services:
- Migration Notes
  - Data Migration Notes
  - Source Code Upgrade Notes
- References

## Summary of Changes from R1 to R2

### Goals

The LRC was significantly redesigned from R1 to R2 with the following goals in mind:

1. Simplification – the old LRC mixed class I concepts such as Result Component with Class II concepts such as Grade and Credits leading to confusion by developers
2. Better ability to logically group values for different purposes – mix and match them
3. Better management of ranges of values by being able to specify the range and increment instead of having to specify all possible values within the range
4. The types were straightened out so they don't mix up multiple concepts

### Objects

1. Result Component has been replaced by Result Values Group
2. Separate objects for each kind of result (grade, credit, credential) have been replaced by a single abstract Result Value that is typed to capture these distinctions.
3. Result Values Groups can now be defined as Result Value Range with a min, max and increment
4. Scale is now Result Scale and can be connected to any type of Result Values Group, previously it was attached to just Grades.

### Types

Even though Result Values Groups essentially replaced Result Component their types are completely different. The old R1 Result Component Types mixed together the two different concepts:

- (1) the type of the results, whether it was a grade or credits and
- (2) the form of the results, whether it was a range or not

So the old list was:

- Certificate
- A single fixed number of credits
- Multiple numbers of credits
- A range of number of credits
- Degree
- GPA
- Final grade

The new R2 Result Values Groups and Result Values have these types separated.

So the Result Values Groups just indicates the form:

- Fixed
- Range
- Multiple

While the new Result Values are tied to a scale whose type indicates the actual type of result:

- Grade
- Admin Grade
- Credit
- Degree
- Minor
- Certification
- Certificate
- Completion
- GPA
- Student Year
- Honor

Note:

- (1) Result Value has a type but for now all there is only one type defined for that object.
- (2) Result Scale really holds the type of the value so perhaps we will revisit this and move that type to the Result Value and not on the scale.

## Methods

1. Removed
  - a. Methods that translate and compare grades – these have been removed pending further review and design
2. Added
  - a. New methods were added to manage (CRUD) the result values (grades, credits) – previously the set of grades or credits could only be added or updated via configuration

## Global Changes to all services:

1. Added context as final parameter to all method calls to support internationalization
2. Made field names consistent, i.e. Descr for all description fields (it was sometimes desc and sometimes descr).
3. Object specific type methods were pulled out into a separate Type Service
4. Object states had simply been strings but were now promoted to interpreted as keys which point to "StateInfo" objects which describe that state and can be internationalized
  - a. State keys were small and in English, i.e. "DRAFT" and now are they are long keys like "kuali.lu.state.draft"
  - b.  This is serious data migration issue which has not yet been resolved  
TODO: create a separate page somewhere for these global changes.

## Migration Notes

### Data Migration Notes

1. Every effort was made to keep the new R2 keys backwards compatible with R1. Therefore the data stored in the CLU resultOption.resultComponentId field in the LU Service should still map to a valid Result Values Group (instead of a Result Component). For example: "kuali.creditType.credit.1.0" which had been the key to a ResultComponent is a valid Result Values Group.
  - See [Google Doc SVCS LRC Types and Values](#)

### Source Code Upgrade Notes

1. The R2 service was called LRCService instead of LrcService. This was a mistake that should probably be fixed (renamed) when copying it back to the lum-api project.
2. CRUD operations on ResultComponent should be changed to work on ResultValuesGroup instead so the object and method calls should be changed
3. from getResultComponent () to getResultValuesGroup ()
4. from createResultComponent () to createResultValuesGroup ()
5. from deleteResultComponent () to deleteResultValuesGroup ()

Not: as far as I can tell these were the only methods used in the R1 LRC contract.

## References

- R2 service [Learning Result Catalog \(LRC\) Service](#)
- R1 Service [Learning Result Catalog Service 1.0](#)

## (CM 2.0) Learning Result Catalog Service Description and Assumptions

### Learning Result Catalog Service

#### Description

The Learning Result Catalog service supports the management of learning results. It provides the basis for defining the result types and values that can be associated with the catalog LU (CLU), the offered LU (LUI), and the specific student (LPR).

#### Key Concepts

##### Result Values Group

- Result Value Group provide a domain of result values, while acting as an abstract result concept. These domains may correspond to scales or simply sets of mutually exclusive achievable results.
- The Result Value Group typically provides context if it maps to a scale, but in other cases, such as credit, the Result Value Group provides little to no additional information, other than acting as a known grouping of Result Values.
- Values associated with Result Values Groups are limited to the same type.

## Result Values

- Result Values serve as a container to the value of a specific result type.

## Specific Result Type

- Specific Results are the foundation of all learning results known by the service.
- The types of results currently defined are credentials, credits, and grades.
- New types of results can be added to this service as needed.
  - *Credentials* correspond to degrees or certificates.
  - *Credits* correspond to an assessment of the amount of time or content. For example, Carnegie Units might be captured using this concept, with the number of units retained within the value field of the creditInfo structure.
  - *Grades* correspond to quantitative assessment of one's progress and mastery of the material covered in the course. Grade values belong to a single scale to allow for easy scale to scale comparisons. This concept is distinct from what might be represented in the Result Value Groups, as the domain of potential grade values may include grade values which are orthogonal from the standard assessment. For example, most current systems have "grades" corresponding to incomplete or withdrawn, which are not true assessments and thus comparisons.

## Assumptions

- Operations to link Result Values and Result Values Groups to other objects are handled in the respective services of record.
- Type setup is part of configuration and not managed specifically through the service, including ResultValuesGroupType.
- Specific Result Value types and values (credentials, credits, grades) are managed in configuration.
- Result Values are reusable entities that can be shared in multiple Result Value Groups.
- A Result Value Group can either have a list of result values or a result value range, but not both.
- A Result Value Range should only have numeric (floating point) values.
- A Result Value for a group with a Result Value Range can be created in respective services of record.

## (CM 2.0) Learning Result Catalog Types and States

These values are used to create Java Constants files to provide programmatic access to this data.

See [LearningResultCatalogServiceConstants.java](#)

## Types

### Result Scale Type

See Google Docs spreadsheet [LRC Types and Values](#)

Key	Name	Description	Status	In Constant File?	In Database?	Comments
kuali.result.scale.type.grade	Grade	The result of an assessment, typically of a course	Proposed			
kuali.result.scale.type.grade.admin	Admin Grade	Grades that are not really the result of assessments but rather indicate administrative issue affecting the assignment of the grade. For example an I for an incomplete grade	Proposed			
kuali.result.scale.type.credit	Credit	Credit that is awarded for completion of a course	Proposed			
kuali.result.scale.type.degree	Degree	Degree awarded for the completion of a particular program of study	Proposed			

kuali.result.scale.type.minor	Minor	Indicates the completion of a smaller area of study in conjunction with a degree	Proposed			
kuali.result.scale.type.certification	Certification	Attestation that a student completed the requirements for a certification for a particular job or profession	Proposed			
kuali.result.scale.type.certificate	Certificate	A certification that a student completed the requirements for a particular area of study that results in a paper certificate being awarded	Proposed			Is a certificate of advanced graduate study really a degree?
kuali.result.scale.type.requirement.completion	Completion	Indicates the completion or lack thereof or the amount of progress towards the completion. Often applied to requirements	Proposed			
kuali.result.scale.type.gpa	GPA	Grade Point Average	Proposed			
kuali.result.scale.type.student.year	Student Year	Student's Year of Study, often within a program i.e. Freshman, Sophomore, Junior, Senior	Proposed			
kuali.result.scale.type.honor	Honor	An honor that is awarded to a student	Proposed			

## Result Values Group Types (formerly Result Component Types)

Key	Name	Description	Status	In Constant File?	In Database?	Comments
kuali.result.value.s.group.type.fixed	Fixed	A result grouping that includes just a single result value	Proposed			
kuali.result.value.s.group.type.range	Range	A grouping that indicates a range of numeric values with a specified increment. It is not necessary that all of the values within that range have been explicitly pre-defined	Proposed			

kuali.result.value.s.group.type.multiple	Multiple	Multiple explicitly named values from which one may be chosen	Proposed			
------------------------------------------	----------	---------------------------------------------------------------	----------	--	--	--

## Result Value Types

Key	Name	Description	Status	In Constant File?	In Database?	Comments
kuali.result.value.type.value	Value	A value	Proposed			not sure if the type key should simply be removed but perhaps we will find something besides this to differentiate

## States

### Result Values Group Life Cycle Process States

Process Key
kuali.result.values.group.process

Key	Name	Next Happy	Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Notes/Comments
kuali.result.values.group.state.draft(initial)	Draft	Approved	The result is just draft and cannot yet be used		Proposed			
kuali.result.values.group.state.approved	Approved		The result has been approved to be used and awarded		Proposed			
kuali.result.values.group.state.retired	Retired		The result has been retired and can still exist on records but can no longer be used		Proposed			

### Result Scale Life Cycle Process States

Process Key
kuali.result.scale.process

Key	Name	Next Happy	Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Notes/Comments
kuali.result.scale.state.draft (initial)	Draft	Approved	The result scale is just draft and cannot yet be used		Proposed			
kuali.result.scale.state.approved	Approved		The result scale has been approved to be used and awarded		Proposed			
kuali.result.scale.state.retired	Retired		The result scale has been retired and can still exist on records but can no longer be used		Proposed			

## Result Value Life Cycle Process States

Process Key
kuali.result.value.process

Key	Name	Next Happy	Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Notes/Comments
kuali.result.value.state.draft (initial)	Draft	Approved	The result value is just draft and cannot yet be used		Proposed			
kuali.result.value.state.approved	Approved		The result value has been approved to be used and awarded		Proposed			
kuali.result.value.state.retired	Retired		The result value has been retired and can still exist on records but can no longer be used		Proposed			

## Existing R1 Data

### Mapping from R1 Result Components

Type Key	Name	Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Notes/Comments
----------	------	-------------	---------	--------------	-------------------	--------------	----------------

kuali.resultComponentType.certificate	Certificate	This records an awarded certificate		Released R1			
kuali.resultComponentType.credit.degree.fixed	Credits, Fixed	This records a single fixed number of credits that are awarded if the student passes the course.		Released R1			
kuali.resultComponentType.credit.degree.multiple	Credits, Multiple	This records multiple numbers of credits that can be awarded for this course.		Released R1			
kuali.resultComponentType.credit.degree.range	Credits, Variable	This records a range of number of credits that can be awarded for this course.		Released R1			
kuali.resultComponentType.degree	Degree	This records a degree is awarded if the student completes the program		Released R1			
kuali.resultComponentType.gpa	Overall GPA	This records the overall GPA of a student in a program		Released R1			
kuali.resultComponentType.grade.finalGrade	Final Grade	This records that a final grade is a result for this course		Released R1			

[More...](#)

## (CM 2.0) Mapping from R1 Result Components

### ▼ [Mapping from R1 Result Components](#)

Type Key	Name	Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Notes/Comments
kuali.resultComponentType.certificate	Certificate	This records an awarded certificate		Released R1			
kuali.resultComponentType.credit.degree.fixed	Credits, Fixed	This records a single fixed number of credits that are awarded if the student passes the course.		Released R1			

kuali.resultComponentType.credit.degree.multiple	Credits, Multiple	This records multiple numbers of credits that can be awarded for this course.		Released R1			
kuali.resultComponentType.credit.degree.range	Credits, Variable	This records a range of number of credits that can be awarded for this course.		Released R1			
kuali.resultComponentType.degree	Degree	This records a degree is awarded if the student completes the program		Released R1			
kuali.resultComponentType.gpa	Overall GPA	This records the overall GPA of a student in a program		Released R1			
kuali.resultComponentType.grade.finalGrade	Final Grade	This records that a final grade is a result for this course		Released R1			

[More...](#)

## (CM 2.0) Result Scale Lifecycle Process States

### Result Scale Life Cycle Process States

Process Key	
kuali.result.scale.process	

Key	Name	Next Happy	Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Notes/Comments
kuali.result.scale.state.draft (initial)	Draft	Approved	The result scale is just draft and cannot yet be used		Proposed			
kuali.result.scale.state.approved	Approved		The result scale has been approved to be used and awarded		Proposed			

kuali.result.scale.state.retired	Retired		The result scale has been retired and can still exist on records but can no longer be used		Proposed			
----------------------------------	---------	--	--------------------------------------------------------------------------------------------	--	----------	--	--	--

## (CM 2.0) Result Scale Types

### Result Scale Type

See Google Docs spreadsheet [LRC Types and Values](#)

Key	Name	Description	Status	In Constant File?	In Database?	Comments
kuali.result.scale.type.grade	Grade	The result of an assessment, typically of a course	Proposed			
kuali.result.scale.type.grade.admin	Admin Grade	Grades that are not really the result of assessments but rather indicate administrative issue affecting the assignment of the grade. For example an I for an incomplete grade	Proposed			
kuali.result.scale.type.credit	Credit	Credit that is awarded for completion of a course	Proposed			
kuali.result.scale.type.degree	Degree	Degree awarded for the completion of a particular program of study	Proposed			
kuali.result.scale.type.minor	Minor	Indicates the completion of a smaller area of study in conjunction with a degree	Proposed			
kuali.result.scale.type.certification	Certification	Attestation that a student completed the requirements for a certification for a particular job or profession	Proposed			

kuali.result.scale.type.certificate	Certificate	A certification that a student completed the requirements for a particular area of study that results in a paper certificate being awarded	Proposed			Is a certificate of advanced graduate study really a degree?
kuali.result.scale.type.requirement.completion	Completion	Indicates the completion or lack thereof or the amount of progress towards the completion. Often applied to requirements	Proposed			
kuali.result.scale.type.gpa	GPA	Grade Point Average	Proposed			
kuali.result.scale.type.student.year	Student Year	Student's Year of Study, often within a program i.e. Freshman, Sophomore, Junior, Senior	Proposed			
kuali.result.scale.type.honor	Honor	An honor that is awarded to a student	Proposed			

## (CM 2.0) Result Types and Structures from R1

### Result Component Type (KSLR\_RESCOMP\_TYPE)

Type Key	Name	Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Notes/Comments
kuali.resultComponentType.certificate	Certificate	This records an awarded certificate		Released R1			
kuali.resultComponentType.credit.degree.fixed	Credits, Fixed	This records a single fixed number of credits that are awarded if the student passes the course.		Released R1			
kuali.resultComponentType.credit.degree.multiple	Credits, Multiple	This records multiple numbers of credits that can be awarded for this course.		Released R1			
kuali.resultComponentType.credit.degree.range	Credits, Variable	This records a range of number of credits that can be awarded for this course.		Released R1			

kuali.resultComponentType.degree	Degree	This records a degree is awarded if the student completes the program		Released R1			
kuali.resultComponentType.gpa	Overall GPA	This records the overall GPA of a student in a program		Released R1			
kuali.resultComponentType.grade.finalGrade	Final Grade	This records that a final grade is a result for this course		Released R1			

### Result Components (KSLR\_RESPCOMP)

▼ SQL code

```
select a.id, a.name, a.type, b.plain
from KSLR_RESPCOMP a,
KSLR_RICH_TEXT_T b
where a.RT_DESCR_ID = b.id (+)
order by type, id
```

Type	Id(key)	Name	Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Notes
kuali.resultComponentType.credit.degree.fixed	kuali.creditType.credit.degree.1.0				Released R1			
kuali.resultComponentType.credit.degree.fixed	kuali.creditType.credit.degree.2.0				Released R1			
kuali.resultComponentType.credit.degree.fixed	kuali.creditType.credit.degree.3.0				Released R1			
kuali.resultComponentType.credit.degree.fixed	kuali.creditType.credit.degree.4.0				Released R1			
kuali.resultComponentType.credit.degree.range	kuali.creditType.credit.degree.1-2				Released R1			
kuali.resultComponentType.credit.degree.range	kuali.creditType.credit.degree.1-3				Released R1			
kuali.resultComponentType.credit.degree.range	kuali.creditType.credit.degree.1-4				Released R1			

kuali.resultComponentType.credit.degeree.range	kuali.creditType.credit.degree.1-6				Released R1			
kuali.resultComponentType.credit.degeree.range	kuali.creditType.credit.degree.3-6				Released R1			
kuali.resultComponentType.degree	kuali.resultComponent.degree.ba	Bachelor of Arts	Bachelor of Arts (BA)		Released R1			
kuali.resultComponentType.degree	kuali.resultComponent.degree.bmus	Bachelor of Music	Bachelor of Music (BSc)		Released R1			
kuali.resultComponentType.degree	kuali.resultComponent.degree.bsc	Bachelor of Science	Bachelor of Science (BSc)		Released R1			
kuali.resultComponentType.grade.fin.alGrade	kuali.resultComponent.grade.audit	Audit	This course can be audited		Released R1			
kuali.resultComponentType.grade.fin.alGrade	kuali.resultComponent.grade.completedNotation	Completed notation	Completed notation		Released R1			
kuali.resultComponentType.grade.fin.alGrade	kuali.resultComponent.grade.designReview	Design review	Design review		Released R1			
kuali.resultComponentType.grade.fin.alGrade	kuali.resultComponent.grade.letter	Letter	Letter, satisfactory		Released R1			
kuali.resultComponentType.grade.fin.alGrade	kuali.resultComponent.grade.passFail	Pass-Fail	This course will have a student selectable pass/fail option		Released R1			
kuali.resultComponentType.grade.fin.alGrade	kuali.resultComponent.grade.percentage	Percentage	Percentage		Released R1			
kuali.resultComponentType.grade.fin.alGrade	kuali.resultComponent.grade.recitalReview	Recital review	Recital review		Released R1			
kuali.resultComponentType.grade.fin.alGrade	kuali.resultComponent.grade.satisfactory	Satisfactory/unsatisfactory	Satisfactory/unsatisfactory		Released R1			

#### Clu Result Type (KSLU\_CLU\_RSLT\_TYP)

Type Key	Name	Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Notes/Comments
----------	------	-------------	---------	--------------	-------------------	--------------	----------------

kuali.resultType.certificate	Certificate	Final learning result for a Certificate Program.		Released R1			
kuali.resultType.creditCourseResult	Final Credits	Final learning result for an LU. A stereotypical usage is the course credits.		Released R1			
kuali.resultType.degree	Degree	Final learning result for a degree		Released R1			
kuali.resultType.gradeCourseResult	Final Grade	Final learning result for an LU. A stereotypical usage is the final grade in a course.		Released R1			

#### Result Usage Type (KSLU\_RSLT\_USG\_TYPE)

⚠ This was never used in R1 – the values for the usage type are all blank!

R2 Type Key	Old R1 Type Key (that was not used)	Name	Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Notes/Comments
kuali.ir.usage.type.final.grade	IrType.finalGrade	Final Grade	Final learning result for an LU. A stereotypical usage is the final grade in a course.		Released R1			
kuali.ir.usage.type.interim.assessment	IrType.midtermGrade	Midterm Grade	Midterm assessment of students progress in a course, typically this is just a flag of those students who are at risk for not doing well in the course.	mid-term grade	Released R1			Changed from R1 to interim instead of mid-term
kuali.ir.usage.type.faculty.submitted.grade		Submitted Grade	Grade as submitted by the faculty before any transformations have been applied, such as converting a letter grade to Pass/Fail.	Original grade, Raw grade, Faculty Assigned Grade	Proposed			

kuali.lr.usag e.type.credit s.earned		Credits Earned	Credits awarded to the student for having successfully passed the course. Typically assign by rule based on the grade but could also be entered directly by the professor to indicate the number of credits earned. This is the raw credits that is used before any additional calculations to see if it should count towards any particular program requirement.		Proposed		
kuali.lr.usag e.type.cred ential.awarde d		Credential Awarded	Credential awarded for having successfully completed the requirements of a program.	Degree awarded	Placeholder		

#### Clu to Result to Result Option to Result Component Id

▼ SQL code

```

select e.LUTYPE_ID, a.TYPE_KEY_ID, b.RES_USAGE_ID, f.type, b.RES_COMP_ID, count (*)
from KSLU_CLU_RSLT a,
     KSLU_CLURES_JN_RESOPT jn,
     KSLU_RSLT_OPT b,
     KSLU_RICH_TEXT_T c,
     KSLU_RICH_TEXT_T d,
     KSLU_CLU e,
     KSLR_RESPCOMP f
where a.id = jn.CLU_RES_ID
and jn.RES_OPT_ID = b.id
and a.clu_id = e.id
and b.res_comp_id = f.id
and b.RT_DESCR_ID = c.id (+)
and a.RT_DESCR_ID = d.id (+)
group by e.LUTYPE_ID, a.TYPE_KEY_ID, b.RES_USAGE_ID, f.type, b.RES_COMP_ID
order by e.LUTYPE_ID, a.TYPE_KEY_ID, b.RES_USAGE_ID, f.type, b.RES_COMP_ID

```

LU Type	Result Type	Result Usage Type	Result Component Type	Result Component ID (should be key?)	Count
kuali.lu.type.CreditCourse	kuali.resultType.creditCourseResult		kuali.resultComponentType.credit.degree.fixed	kuali.creditType.credit.degree.1.0	23
kuali.lu.type.CreditCourse	kuali.resultType.creditCourseResult		kuali.resultComponentType.credit.degree.fixed	kuali.creditType.credit.degree.2.0	14
kuali.lu.type.CreditCourse	kuali.resultType.creditCourseResult		kuali.resultComponentType.credit.degree.fixed	kuali.creditType.credit.degree.3.0	422
kuali.lu.type.CreditCourse	kuali.resultType.creditCourseResult		kuali.resultComponentType.credit.degree.fixed	kuali.creditType.credit.degree.4.0	59
kuali.lu.type.CreditCourse	kuali.resultType.creditCourseResult		kuali.resultComponentType.credit.degree.range	kuali.creditType.credit.degree.1-2	2
kuali.lu.type.CreditCourse	kuali.resultType.creditCourseResult		kuali.resultComponentType.credit.degree.range	kuali.creditType.credit.degree.1-3	8
kuali.lu.type.CreditCourse	kuali.resultType.creditCourseResult		kuali.resultComponentType.credit.degree.range	kuali.creditType.credit.degree.1-4	3
kuali.lu.type.CreditCourse	kuali.resultType.creditCourseResult		kuali.resultComponentType.credit.degree.range	kuali.creditType.credit.degree.1-6	5
kuali.lu.type.CreditCourse	kuali.resultType.creditCourseResult		kuali.resultComponentType.credit.degree.range	kuali.creditType.credit.degree.3-6	3
kuali.lu.type.CreditCourse	kuali.resultType.gradeCourseResult		kuali.resultComponentType.grade.finalGrade	kuali.resultComponent.grade.audit	491
kuali.lu.type.CreditCourse	kuali.resultType.gradeCourseResult		kuali.resultComponentType.grade.finalGrade	kuali.resultComponent.grade.letter	537
kuali.lu.type.CreditCourse	kuali.resultType.gradeCourseResult		kuali.resultComponentType.grade.finalGrade	kuali.resultComponent.grade.passFail	499
kuali.lu.type.CreditCourse	kuali.resultType.gradeCourseResult		kuali.resultComponentType.grade.finalGrade	kuali.resultComponent.grade.satisfactory	2
kuali.lu.type.MajorDiscipline	kuali.resultType.degree		kuali.resultComponentType.degree	kuali.resultComponent.degree.ba	1
kuali.lu.type.MajorDiscipline	kuali.resultType.degree		kuali.resultComponentType.degree	kuali.resultComponent.degree.bsc	2
kuali.lu.type.Variation	kuali.resultType.degree		kuali.resultComponentType.degree	kuali.resultComponent.degree.bsc	2
kuali.lu.type.credental.Baccalaureate	kuali.resultType.degree		kuali.resultComponentType.degree	kuali.resultComponent.degree.ba	1

kuali.lu.type.credentials.Baccalaureate	kuali.resultType.degree		kuali.resultComponentType.degree	kuali.resultComponentType.degree.bmus	1
kuali.lu.type.credentials.Baccalaureate	kuali.resultType.degree		kuali.resultComponentType.degree	kuali.resultComponentType.degree.bsc	1

## Result Component Id to Result Component Values

### SQL code

```
select a.id, a.type, a.name, b.value
from KSLR_RESPCOMP a,
KSLR_RESULT_VALUE b
where a.id = b.RSLT_COMP_ID
order by a.id, b.value
```

Result Component Id (should be key?)	Result Component Type	Result Component Name	values
kuali.creditType.credit.degree.1-2	kuali.resultComponentType.credit.degree.range		1
kuali.creditType.credit.degree.1-2	kuali.resultComponentType.credit.degree.range		2
kuali.creditType.credit.degree.1-3	kuali.resultComponentType.credit.degree.range		1
kuali.creditType.credit.degree.1-3	kuali.resultComponentType.credit.degree.range		2
kuali.creditType.credit.degree.1-3	kuali.resultComponentType.credit.degree.range		3
kuali.creditType.credit.degree.1-4	kuali.resultComponentType.credit.degree.range		1
kuali.creditType.credit.degree.1-4	kuali.resultComponentType.credit.degree.range		2
kuali.creditType.credit.degree.1-4	kuali.resultComponentType.credit.degree.range		3
kuali.creditType.credit.degree.1-4	kuali.resultComponentType.credit.degree.range		4
kuali.creditType.credit.degree.1-6	kuali.resultComponentType.credit.degree.range		1
kuali.creditType.credit.degree.1-6	kuali.resultComponentType.credit.degree.range		2
kuali.creditType.credit.degree.1-6	kuali.resultComponentType.credit.degree.range		3
kuali.creditType.credit.degree.1-6	kuali.resultComponentType.credit.degree.range		4
kuali.creditType.credit.degree.1-6	kuali.resultComponentType.credit.degree.range		5
kuali.creditType.credit.degree.1-6	kuali.resultComponentType.credit.degree.range		6

kuali.creditType.credit.degree.3-6	kuali.resultComponentType.credit.degree.range		3
kuali.creditType.credit.degree.3-6	kuali.resultComponentType.credit.degree.range		4
kuali.creditType.credit.degree.3-6	kuali.resultComponentType.credit.degree.range		5
kuali.creditType.credit.degree.3-6	kuali.resultComponentType.credit.degree.range		6
kuali.creditType.credit.degree.1-0	kuali.resultComponentType.credit.degree.fixed		1
kuali.creditType.credit.degree.2-0	kuali.resultComponentType.credit.degree.fixed		2
kuali.creditType.credit.degree.3-0	kuali.resultComponentType.credit.degree.fixed		3
kuali.creditType.credit.degree.4-0	kuali.resultComponentType.credit.degree.fixed		4
kuali.resultComponent.grade.completedNotation	kuali.resultComponentType.grade.finalGrade	Completed notation	Completed
kuali.resultComponent.grade.completedNotation	kuali.resultComponentType.grade.finalGrade	Completed notation	In-Progress
kuali.resultComponent.grade.completedNotation	kuali.resultComponentType.grade.finalGrade	Completed notation	Not-Completed
kuali.resultComponent.grade.letter	kuali.resultComponentType.grade.finalGrade	Letter	A
kuali.resultComponent.grade.letter	kuali.resultComponentType.grade.finalGrade	Letter	B
kuali.resultComponent.grade.letter	kuali.resultComponentType.grade.finalGrade	Letter	C
kuali.resultComponent.grade.letter	kuali.resultComponentType.grade.finalGrade	Letter	D
kuali.resultComponent.grade.letter	kuali.resultComponentType.grade.finalGrade	Letter	F
kuali.resultComponent.grade.passFail	kuali.resultComponentType.grade.finalGrade	Pass-Fail	Fail
kuali.resultComponent.grade.passFail	kuali.resultComponentType.grade.finalGrade	Pass-Fail	Pass
kuali.resultComponent.grade.percentage	kuali.resultComponentType.grade.finalGrade	Percentage	0-59%
kuali.resultComponent.grade.percentage	kuali.resultComponentType.grade.finalGrade	Percentage	60-69%
kuali.resultComponent.grade.percentage	kuali.resultComponentType.grade.finalGrade	Percentage	70-79%

kuali.resultComponent.grade.percentage	kuali.resultComponentType.grade.finalGrade	Percentage	80-84%
kuali.resultComponent.grade.percentage	kuali.resultComponentType.grade.finalGrade	Percentage	85-89%
kuali.resultComponent.grade.percentage	kuali.resultComponentType.grade.finalGrade	Percentage	90-94%
kuali.resultComponent.grade.percentage	kuali.resultComponentType.grade.finalGrade	Percentage	>95%
kuali.resultComponent.grade.satisfactory	kuali.resultComponentType.grade.finalGrade	Satisfactory/unsatisfactory	Not-Satisfactory
kuali.resultComponent.grade.satisfactory	kuali.resultComponentType.grade.finalGrade	Satisfactory/unsatisfactory	Satisfactory

## (CM 2.0) Result Value Group Types

### Result Values Group Types (formerly Result Component Types)

Key	Name	Description	Status	In Constant File?	In Database?	Comments
kuali.result.value.s.group.type.fixed	Fixed	A result grouping that includes just a single result value	Proposed			
kuali.result.value.s.group.type.range	Range	A grouping that indicates a range of numeric values with a specified increment. It is not necessary that all of the values within that range have been explicitly pre-defined	Proposed			
kuali.result.value.s.group.type.multiple	Multiple	Multiple explicitly named values from which one may be chosen	Proposed			

## (CM 2.0) Result Value Lifecycle Process States

### Result Value Life Cycle Process States

Process Key
kuali.result.value.process

Key	Name	Next Happy	Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Notes/Comments

kuali.result.value.state.draft (initial)	Draft	Approved	The result value is just draft and cannot yet be used		Proposed			
kuali.result.value.state.approved	Approved		The result value has been approved to be used and awarded		Proposed			
kuali.result.value.state.retired	Retired		The result value has been retired and can still exist on records but can no longer be used		Proposed			

## (CM 2.0) Result Values Group Lifecycle Process States

### Result Values Group Life Cycle Process States

Process Key								
kuali.result.values.group.process								

Key	Name	Next Happy	Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Notes/Comments
kuali.result.values.group.state.draft (initial)	Draft	Approved	The result is just draft and cannot yet be used		Proposed			
kuali.result.values.group.state.approved	Approved		The result has been approved to be used and awarded		Proposed			
kuali.result.values.group.state.retired	Retired		The result has been retired and can still exist on records but can no longer be used		Proposed			

## (CM 2.0) Result Value Types

### Result Value Types

Key	Name	Description	Status	In Constant File?	In Database?	Comments
-----	------	-------------	--------	-------------------	--------------	----------

kuali.result.value.type.value	Value	A value	Proposed			 not sure if the type key should simply be removed but perhaps we will find something besides this to differentiate
-------------------------------	-------	---------	----------	--	--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## (CM 2.0) Learning Result Catalog Gordon Documentation

- Overview
- [KSEN\\_LRC\\_RVG](#)
  - Columns
    - ID
    - RVG\_TYPE
    - ...
- [KSEN\\_RESULT\\_VALUE](#)
  - Columns
    - ID
    - RESULT\_SCALE\_ID
    - RESULT\_VALUE
    - ...

### Overview

The Learning Result Catalog (LRC) Service manages pretty much all of information pertaining to grades, credits and credentials. The three primary tables are KSEN\_LRC\_RVG, KSEN\_LRC\_RVG\_RESULT\_VALUE, and KSEN\_RESULT\_VALUE.KSEN\_LRC\_RVG defines all of the various Learning Results: credits awarded, GPAs, letter grades, percentage grades, degrees, etc.

KSEN\_LRC\_RVG defines individual Result Value Groups, like credit options and grading options.

KSEN\_LRC\_RVG\_RESULT\_VALUE joins Learning Result Groups with all possible values/outcomes for that group.

KSEN\_RESULT\_VALUE distinctly defines all possible results/outcomes. If two or more Result Value Groups share a common Result Value, they will all reference the same Result Value record.

KSEN\_RESULT\_SCALE defines categories of RVG and RV records.

### [KSEN\\_LRC\\_RVG](#)

This table contains the Result Value Group (RVG) records. There are many categories of RVG records, like credit options and grading options; categories are identified using the ID column.

#### Columns

##### **ID**

The ID column is comprised of a generated prefix and a generated suffix; it is not a GUID.

The prefix is a value based on the category of the RVG itself. Credit options are identified by the prefix "kuali.creditType.credit.degree".

The suffix is formulated using naming conventions that are particular to each category, but serve to uniquely identify each record within the category. For credit options, this naming convention involves the use of the min and max values, as well as the increment value, optimally omitting redundant or missing values; the suffix for the RVG for 3.0 credits would be "3.0", for 3.0-5.0 credits would be "3.0-5.0" and for 3.0-4.0 credits in .5 credit increments would be "3.0-5.0.by..5".

##### **RVG\_TYPE**

This column identifies the characteristic of the Result Values (RV) associated with this RVG. There are three possible values for this column: kuali.result.values.group.type.fixed, kuali.result.values.group.type.range, and kuali.result.values.group.type.multiple.

Fixed type RVGs have 1 and only 1 associated RV. Fixed type RVGs have the same value for RANGE\_MIN\_VALUE and RANGE\_MAX\_VALUE, and should have null for the RANGE\_INCREMENT value.

Ranged type RVGs have 1 or more associated RVs. Ranged type RVGs use the RANGE\_MIN\_VALUE and RANGE\_MAX\_VALUE columns to determine the lower and upper limits to the range, and use the RANGE\_INCREMENT column to determine the actual values associated with the RVG.

Multiple type RVGs have 1 or more associated RVs. Multiple type RVGs do not use the RANGE columns because they represent a fixed set of RVs that do not necessarily form a numeric range.

...

The remainder of the columns are fairly self-explanatory.

## KSEN\_RESULT\_VALUE

This table distinctly identifies all possible Result Value (RV) records associated with RVGs. If two or more RVGs refer to an identical RV, they will reference the same RV record, so there is a many-to-one relationship between KSEN\_LRC\_RVG and KSEN\_RESULT\_VALUE (which is a change from how 1.0 was implemented as a one-to-one).

### Columns

#### *ID*

This column uniquely identifies the Result Value record. The value is generated; the logic used to determine the value depends on the particular RESULT\_SCALE. "Grade" type result scales use a value that represents that particular grading result; the values of the ID column and RESULT\_VALUE column are identical. "credit.degree" type result scale uses a generated key with a prefix and suffix. The prefix is "kuali.result.value.credit.degree". The suffix comes from the RESULT\_VALUE column.

#### **RESULT\_SCALE\_ID**

Identifies the particular result scale category to which this result value belongs. Presumably, distinct categories of result scales will never have identical RESULT\_VALUE values, so there should be no problem with conflicting IDs.

#### **RESULT\_VALUE**

Identifies the value being represented by the RV record. Commonly used to formulate the ID value. This is the column to use when working with the values, for example when generating a report card or calculating a GPA.

...

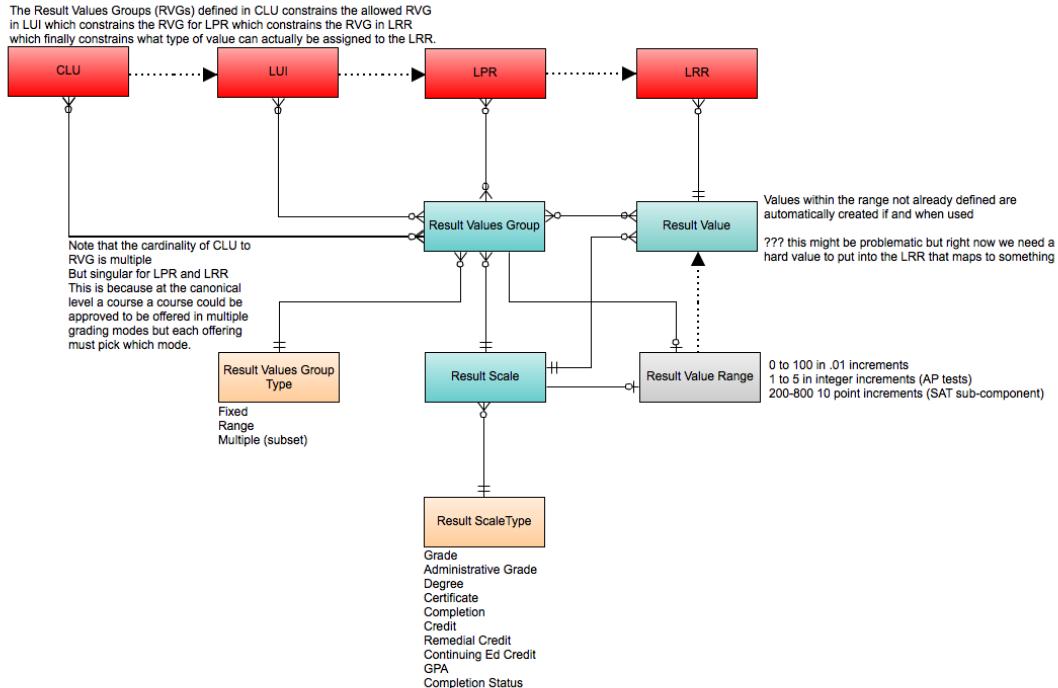
The remainder of the columns are fairly self-evident and self-explanatory.

## (CM 2.0) Learning Result Catalog Service Entity Diagram

- [Entity Model](#)
- [Data Model and DDL](#)

### Entity Model

## Learning Result Catalog Service Entity Diagram



## Data Model and DDL

[KSEN\\_LRC.pdf](#)  
[KSEN\\_LRC\\_DDL\\_TABLES.ddl](#)  
[KSEN\\_LRC\\_DDL\\_CONSTRAINTS.ddl](#)

## (CM 2.0) Message Service

- Class
- Description
- Entity Diagram
- Service Contract Documentation
- Type and State Configuration

## Class

Message Service is a Core service that stores information about a message or label.

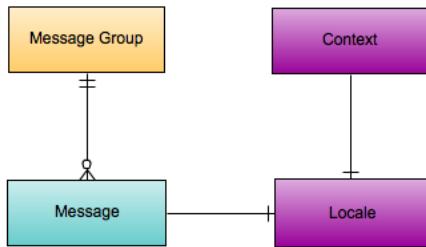
## Description

Manages locale specific messages for internationalization. A *message* is unique in the context of *locale* and *group*. The *locale* parameter allows for operations pertaining to messages whose locale is different from that of *ContextInfo* locale information.

[More...](#)

## Entity Diagram

## Message Service Entity Diagram



[More...](#)

## Service Contract Documentation

[Go to Message Service Contract Documentation](#)

## Type and State Configuration

Group Key	Name	Description	Status	Notes/Comments
org	Organization Messages		R1	
program	Program Messages		R1	
common	Common Messages		R1	
course	Course Messages		R1	
validation	Validation Messages		R1	
clusetmanagement	Clu Set Management Messages		R1	

Note: These do not follow the standard type prefix of "kuali.message.group.type.xxxx" because they were never implemented that way in R1

[More...](#)

## (CM 2.0) Message Service Description and Assumptions

### Message Service

#### Description

Manages locale specific messages for internationalization. A *message* is unique in the context of *locale* and *group*. The *locale* parameter allows for operations pertaining to messages whose locale is different from that of *ContextInfo* locale information.

#### Key Concepts

- MessageGroupKey is used to group messages into categories such as business concepts, organization, program, course, or overall categories such as common and validation.
- Locale is part of contextInfo which is specified as part of virtually every service call.

## (CM 2.0) Message Service Types and States

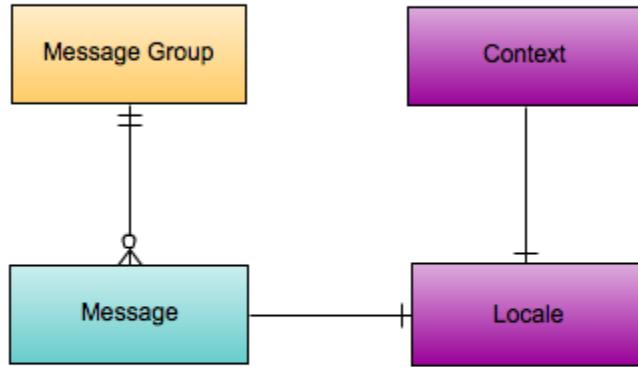
Group Key	Name	Description	Status	Notes/Comments
-----------	------	-------------	--------	----------------

org	Organization Messages		R1	
program	Program Messages		R1	
common	Common Messages		R1	
course	Course Messages		R1	
validation	Validation Messages		R1	
clusetmanagement	Clu Set Management Messages		R1	

Note: These do not follow the standard type prefix of "kuali.message.group.type.xxxx" because they were never implemented that way in R1

## (CM 2.0) Message Service Entity Diagram

### Message Service Entity Diagram



## (CM 2.0) Organization Service

- Class
- Description
- Entity Diagram
- Service Contract Documentation
- Type and State Configuration

### Class

Organization Service is a Core service that manages organizational.

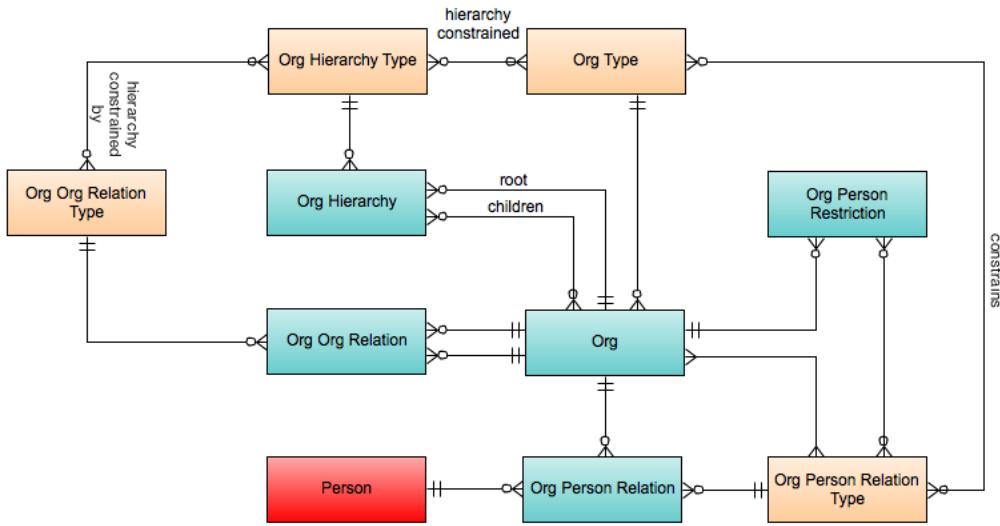
### Description

This service manages organizational units that have a relationship to the institution. The organizations may be internal and include officially recognized organizations (e.g. Departments, Faculties, Schools) or unofficial organizations (e.g. clubs or student groups), or they may be external organizations (e.g. companies, other institutions, government, associations). This service also manages the relationships between people and organizations.

More...

## Entity Diagram

### Organization Service Entity Diagram



[More...](#)

## Service Contract Documentation

[Go to Organization Service Contract Documentation](#)

## Type and State Configuration

[More...](#)

## (CM 2.0) Organization Service Description and Assumptions

### Organization Service

#### Description

This service manages organizational units that have a relationship to the institution. The organizations may be internal and include officially recognized organizations (e.g. Departments, Faculties, Schools) or unofficial organizations (e.g. clubs or student groups), or they may be external organizations (e.g. companies, other institutions, government, associations). This service also manages the relationships between people and organizations.

#### Key Concepts

- Organizations are distinguished from **authorization groups** in that organizations deal directly with people while groups deal directly with principals. In other words, organizations may be comprised of individuals who have no way to authenticate themselves (and thus have no unique permissions) and AZ groups may have principals which are linked to non-human entities (such as batch jobs, other services, etc.).
- Organizations and groups may be related, in that a member of an organization may have one of their principals associated with an AZ group, but this is not required.

#### Assumptions

- Most organizations have "parent" organization(s) within a given context, e.g., the School of Arts and Sciences exists within the institution as a whole. Parent:child:institution:School of Arts and Sciences
- The "parent" organization(s) of an existing organization may shift depending on the context. This leads to the need to capture multiple

relationships for a given organization, e.g., a department may report to a particular institution for administrative purposes, but report to another institution for financial purposes.

- Organization to organization relationships can be grouped into hierarchies based upon the type of relationship.
- Organizations may place additional constraints on the types of relationships a person may have with the organization.

## (CM 2.0) Organization Service Types and States

### Types

Organization Types - these are from CM and need to be updated to reflect the naming convention::  
kuali.organization.type.XXX (all lowercase)

Type Key	Name	Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Notes/Comments
kuali.org.CorporateEntity	Corporate Entity			Placeholder	N	N	These were used in CM and updated with UW Ref Data. UMD is currently looking at all of these and will provide their feedback as Implementing Institution. After feedback and review by Core Analysis, Melissa will update this table to reflect the type naming convention for ENR and create a Jira to assign to Sri for configuration in CF and Charles for configuration in database.
kuali.org.Office	Office			Placeholder	N	N	
kuali.org.Division	Division			Placeholder	N	N	
kuali.org.College	College			Approved	N	N	 <b>KSENR</b> <b>066</b> - Update Organization Ty  <b>Close</b>
kulai.org.type.department	Department			Placeholder	N	N	
kuali.org.Senate	Senate			Placeholder	N	N	
kuali.org.Program	Program			Placeholder	N	N	
kuali.org.School	School			Placeholder	N	N	

kuali.org.Center	Center			Placeholder	N	N	
kuali.org.TestingService	Testing Service			Placeholder	N	N	
kuali.org.Committee	Committee			Placeholder	N	N	
kuali.org.Association	Association			Placeholder	N	N	
kuali.org.AdvisoryGroup	Advisory Group			Placeholder	N	N	
kuali.org.Department	Department			Approved	N	N	 
kuali.org.Board	Board			Placeholder	N	N	
kuali.org.type.college	College			Placeholder	N	N	
kuali.org.COC	COC			Placeholder	N	N	
kuali.org.WorkGroup	Work Group			Placeholder	N	N	
kuali.org.Section	Section			Placeholder	N	N	
kuali.org.type.university	University			Placeholder	N	N	
kuali.org.AccreditingBody	Accrediting Body			Placeholder	N	N	
kuali.org.AdhocCommittee	Ad Hoc Committee			Placeholder	N	N	
kuali.org.type.campus	Campus			Placeholder	N	N	

## (CM 2.0) Organization Types

Organization Types - these are from CM and need to be updated to reflect the naming convention::  
kuali.organization.type.XXX (all lowercase)

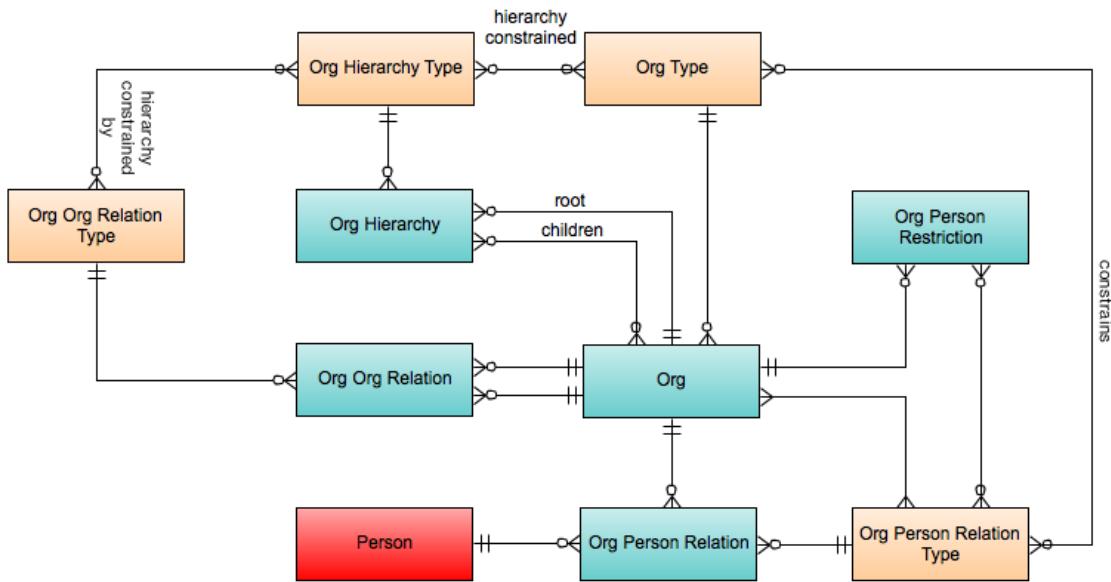
Type Key	Name	Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Notes/Comments

kuali.org.CorporateEntity	Corporate Entity			Placeholder	N	N	These were used in CM and updated with UW Ref Data. UMD is currently looking at all of these and will provide their feedback as Implementing Institution. After feedback and review by Core Analysis, Melissa will update this table to reflect the type naming convention for ENR and create a Jira to assign to Sri for configuration in CF and Charles for configuration in database.
kuali.org.Office	Office			Placeholder	N	N	
kuali.org.Division	Division			Placeholder	N	N	
kuali.org.College	College			Approved	N	N	 KSENR 066 - Update Organization Type 
kulai.org.type.department	Department			Placeholder	N	N	
kuali.org.Senate	Senate			Placeholder	N	N	
kuali.org.Program	Program			Placeholder	N	N	
kuali.org.School	School			Placeholder	N	N	
kuali.org.Center	Center			Placeholder	N	N	
kuali.org.TestingService	Testing Service			Placeholder	N	N	
kuali.org.Committee	Committee			Placeholder	N	N	
kuali.org.Association	Association			Placeholder	N	N	
kuali.org.AdvisoryGroup	Advisory Group			Placeholder	N	N	

kuali.org.Department	Department			Approved	N	N	<input checked="" type="checkbox"/> KSENR 066 - Update Organization Ty 
kuali.org.Board	Board			Placeholder	N	N	
kuali.org.type.college	College			Placeholder	N	N	
kuali.org.COC	COC			Placeholder	N	N	
kuali.org.Work Group	Work Group			Placeholder	N	N	
kuali.org.Section	Section			Placeholder	N	N	
kuali.org.type.university	University			Placeholder	N	N	
kuali.org.AccreditingBody	Accrediting Body			Placeholder	N	N	
kuali.org.AdhocCommittee	Ad Hoc Committee			Placeholder	N	N	
kuali.org.type.campus	Campus			Placeholder	N	N	

## (CM 2.0) Organization Service Entity Diagram

Organization Service Entity Diagram



## (CM 2.0) Program Service

- Class
- Description
- Entity Diagram
- Layering Diagram
- Service Contract Documentation
- Types and States

## Class

The Program Service is a Class II service that manages program creation and approval.

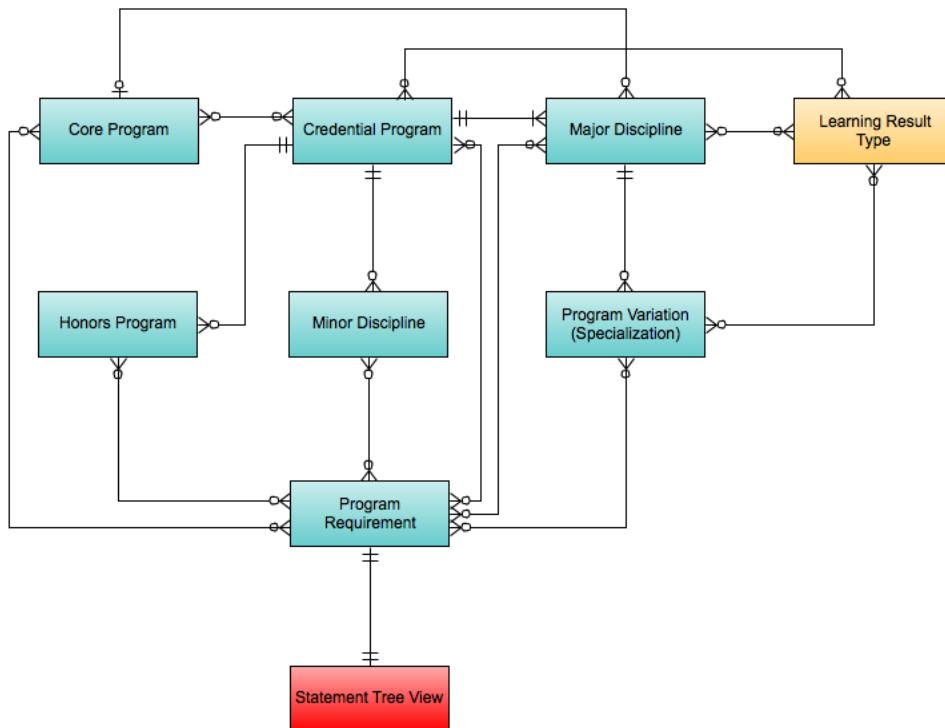
## Description

A program is a prescribed grouping of learning units such as courses, activities, competencies, learning objectives, and/or institutional/core requirements that lead to a recognized body of knowledge. Programs may also include requirements as to the order, timing, performance, and results (e.g. GPA) of the grouping of learning units that make up a program. The end result of a completed program may be an acknowledged level of accomplishment (e.g. a major, minor or concentration) or a credential (e.g. certificate, degree, etc.)

[More...](#)

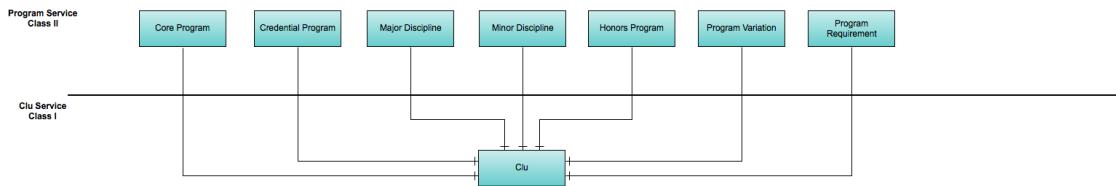
## Entity Diagram

**Program Service Entity Diagram**



[More...](#)

## Layering Diagram



More...

## Service Contract Documentation

[Go to Program Service Contract Documentation](#)

## Types and States

More...

## (CM 2.0) Program Service Description and Assumptions

### Program Service

#### Description

A program is a prescribed grouping of learning units such as courses, activities, competencies, learning objectives, and/or institutional/core requirements that lead to a recognized body of knowledge. Programs may also include requirements as to the order, timing, performance, and results (e.g. GPA) of the grouping of learning units that make up a program. The end result of a completed program may be an acknowledged level of accomplishment (e.g. a major, minor or concentration) or a credential (e.g. certificate, degree, etc.)

The goal for Kuali Student programs is to capture the program information in a structured format that can feed both a degree audit application on one side and a published course catalog on the other side. It represents a Curriculum (provost) and curriculum manager view of the Program. Rules for programs serve to identify the relationships between courses and programs and provides the basis for dependency analysis.

#### Key Concepts

##### Credential Program

A classification of canonical learning units (types) that lead to some sort of credential. Here are some of the types of credential programs.

- Baccalaureate Program is an organized curriculum leading to a baccalaureate degree in a major area of study. Baccalaureate programs are typically composed of a major area of study, common institution/school requirements (general education), and elective coursework/study. Different institutions will have different structures/components.
- Masters Program is a postgraduate academic program typically designed for learners after the completion of a baccalaureate program. Masters programs have varying levels of individualization ranging from highly regimented programs with specific course requirements to programs whose requirements are determined by the learner's specific goals. These programs typically include a thesis and/or comprehensive examination. Graduate programs are typically differentiated as professional or research-based programs.
- Doctoral Program is a postgraduate academic program typically representing the highest level of study in a field (in a few fields a "terminal" Master's degree may be awarded, such as the Masters of Fine Arts). Doctoral programs typically range from 5 to 7 years, of which a considerable portion (and in some cases all) of the work takes place outside of traditional course settings.

##### Major Discipline

- An organized curriculum offered as a major area of study recognized as an academic discipline by the higher education community as part of a baccalaureate degree program.
- A learner may complete more than one major area of study within a single baccalaureate program.
- Majors are managed as entities, by departments, outside of the related **Credential Program**.

##### Minor Discipline

- Any organized curriculum that is offered as part of an individual student's degree program and which enhances or complements the degree to be awarded in a manner that leads to specific educational or occupational goals.
- The credit hour length is set in accordance with institutional policy and typically requires less coursework/study than a major area of study.

##### Program Variations

- Variations of a major area of study that lead to specific educational or occupational goals, also known as Specializations, Concentrations, Areas of Emphasis, Tracks, etc.
- Some variations may represent a portion of unique curricula in relation to the major area of study, while others specify more focused paths of study within the boundaries of a major area of study.

#### **General Education or Core Program**

- An organized grouping of courses that is required for all students as a foundation of general knowledge.
- The courses typically span a breadth of knowledge in areas such as Mathematics, English Composition, History/Social Science, Humanities and Fine Arts, and Natural Sciences.

#### **Honors Program**

- Departments often have Honors programs that are managed at the department level and are available to the major programs that department offers.

#### **Certificate Program**

See *Credential Program*.

#### **Program Requirement**

- Programs are typically defined by the "requirements" which describe program goals and the courses that can meet those goals.
- These are defined in CM as "rules" which are grouped by statement type to define requirements for Entrance, Satisfactory Progress and Completion.

#### **Assumptions**

- Type setup is part of configuration and not managed specifically through the service, including learnignResultType, programType
- At least one major area of study must be completed as part of a baccalaureate program.
- The completion of General Education Core along with an undergraduate program/major is often a requirement of a baccalaureate program.

#### **Deferred Design**

- Structured programs
  - Time (atpType, generic term1, etc.) as part of Program Requirements

## **(CM 2.0) Program Service Types and States**

### **Program Types**

Type Key	Name	Description	Aliases	Agreed Upon	In Constant File?	In Database?	Notes/Comments
kuali.lu.type.credential.Baccalaureate	Baccalaureate	A Baccalaureate		Placeholder			
kuali.lu.type.credential.Masters	Masters	Masters level program		Placeholder			
kuali.lu.type.credential.Professional	Professional	A program leading to a professional degree		Placeholder			
kuali.lu.type.CoreProgram	Core	Program containing core requirements		Placeholder			
kuali.lu.type.MajorDiscipline	Major	A Major Discipline		Placeholder			
kuali.lu.type.Variation	Specialization	Specialization of a Major Discipline		Placeholder			

kuali.lu.type.Requirement	Req	Program requirements		Placeholder			
---------------------------	-----	----------------------	--	-------------	--	--	--

## (CM 2.0) Program Types

### Program Types

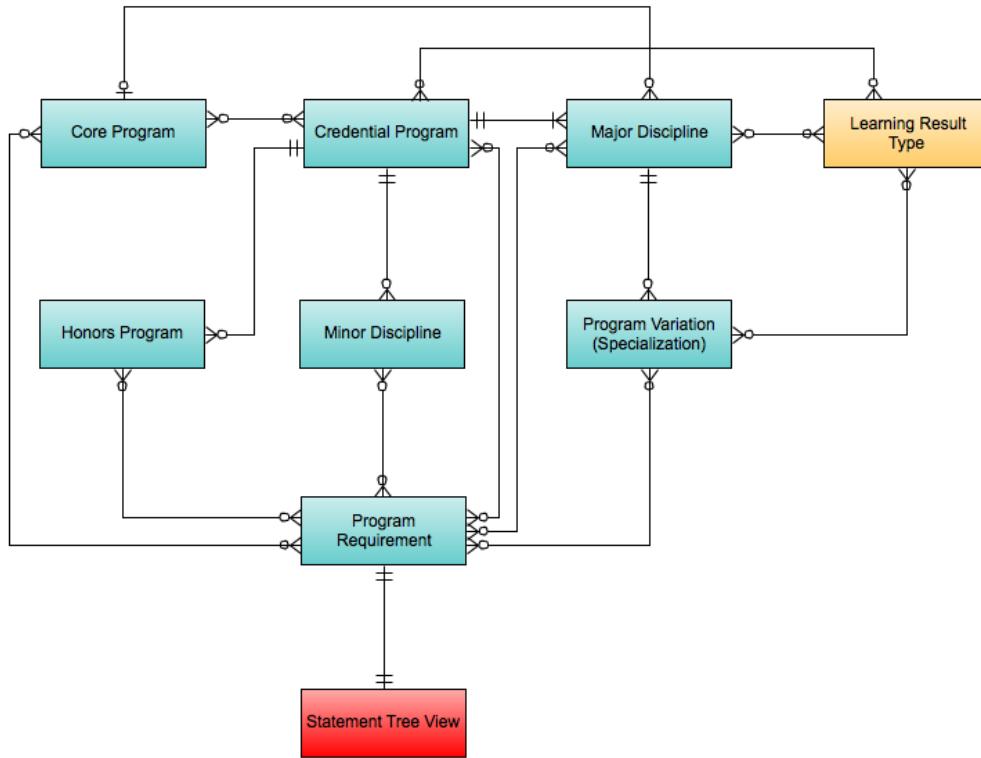
Type Key	Name	Description	Aliases	Agreed Upon	In Constant File?	In Database?	Notes/Comments
kuali.lu.type.credential.Baccalaureate	Baccalaureate	A Baccalaureate		Placeholder			
kuali.lu.type.credential.Masters	Masters	Masters level program		Placeholder			
kuali.lu.type.credential.Professional	Professional	A program leading to a professional degree		Placeholder			
kuali.lu.type.CoreProgram	Core	Program containing core requirements		Placeholder			
kuali.lu.type.MajorDiscipline	Major	A Major Discipline		Placeholder			
kuali.lu.type.Variation	Specialization	Specialization of a Major Discipline		Placeholder			
kuali.lu.type.Requirement	Req	Program requirements		Placeholder			

## (CM 2.0) Program Service Entity Diagram

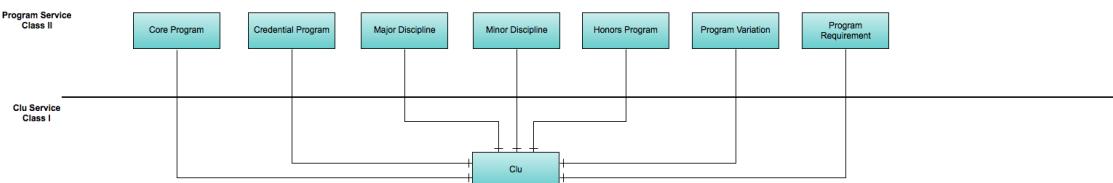
- [Entity Diagram](#)
- [Layering Diagram](#)

### Entity Diagram

## Program Service Entity Diagram



## Layering Diagram



## (CM 2.0) Proposal Service

- Class
- Description
- Entity Diagram
- Service Contract Documentation
- Type and State Configuration
  - Proposal Types
  - Reference Types

## Class

Proposal is a Class I service supports the management of proposals.

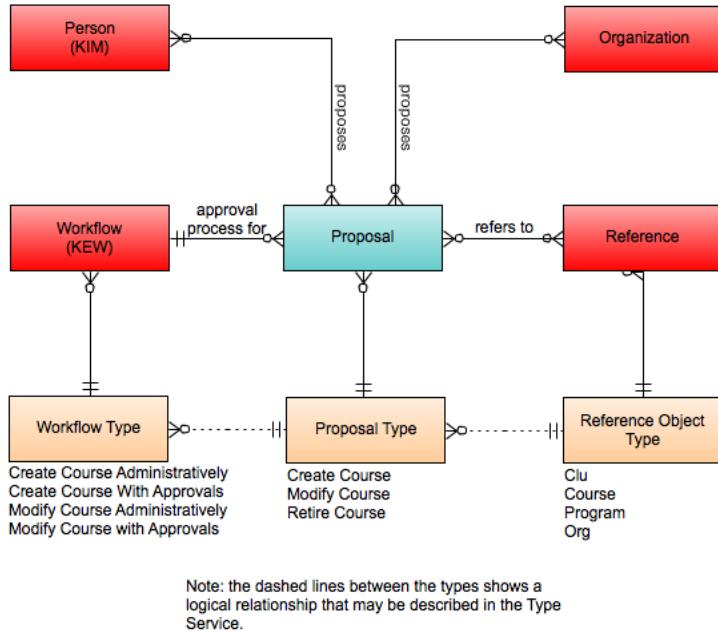
## Description

The Proposal Service supports the management of Proposals. The proposal is general and can be associated with any arbitrary entity in the system, for example, a learning unit. Proposals have types that capture the activity being proposed, for example, Creating a new course or Updating the prefix for a set of courses or Launching a new student club.

[More...](#)

## Entity Diagram

**Proposal Service Entity Diagram**



[More...](#)

## Service Contract Documentation

[Go to Proposal Service Contract Documentation](#)

## Type and State Configuration

### Proposal Types

Key	Name	Description	Status	In Constant File?	In Database?	Notes/Comments
kuali.proposal.type.course.create	Create Course Proposal	A create course proposal type	Configured in R1			
kuali.proposal.type.course.modify	Modify Course Proposal	A modify course proposal type	Configured in R1			

kuali.proposal.type.course.retire	Retire Course Proposal	A retire course proposal type	Configured in R1			
-----------------------------------	------------------------	-------------------------------	------------------	--	--	--

## Reference Types

Key	Name	Description	Status	In Constant File?	In Database?	Notes/Comments
kuali.proposal.referenceType.clu	Clu Proposal Reference	Clu proposal reference type	Configured in R1			This should be a Reference Object Type Key (refObjectUri)

More...

## (CM 2.0) Proposal Service Description and Assumptions

### Proposal Service

#### Description

The Proposal Service supports the management of Proposals. The proposal is general and can be associated with any arbitrary entity in the system, for example, a learning unit. Proposals have types that capture the activity being proposed, for example, Creating a new course or Updating the prefix for a set of courses or Launching a new student club.

A proposal may reference more than one (1) object, provided that the objects are all of the same reference type and that the other data contained in the proposal applies to all of the objects referenced. An object (e.g. CLU) is involved in **only one** active proposal at a time.

### Key Concepts

#### ProposalTypes

- Proposal types provide the basic structure or template for creating the Proposal, e.g., CreateNewCourse, CourseCorrection.
- Depending on the type, there may be different information associated with the Proposal. Examples of Proposal Types include Course Correction, Course Modification, etc.
- The granularity or Proposal Types depends on the implementation.
- The type of the Proposal may also indicate the type of CRUD activity and have implications for the associated object. For a proposalType of CourseCorrection, the referenced object is an existing cluld; for a proposalType of CreateNewCourse, it would be a new cluld.
- ProposalType is constrained by referenceType.

#### ReferenceType

- Reference type is overall type of the Proposal e.g., CreateNewCourse, CourseCorrection.
- Reference type along with an ID, reflects the specific entity associated with the proposal. For example, a Proposal may reference an object of type CLU along with its CLU Id. The referenceTypKey and the referenceId concepts are also found in the [Comment Service](#).

#### ProposalDocRelationType

- Propsoal document relation types reflect the relation type between a proposal and a document, e.g., proposal rationale, syllabus, etc.
- While it could be similar to the luDocRelationType, ultimately these are two distinct concepts. It is also distinct from what the documentTypeKey which captures the file format, e.g., PDF.

### Assumptions

- A proposal contains **no** codified information about the change, though it is present in the description of the desired change. Instead the Proposal information is very generic and relies on the associated entity for specific information.
- Documents can be attached to a proposal as needed to support the "proposed" activity.
- The proposal contains a reference to what it is associated with, e.g., clulD, orglD, etc.
- The object referenced by the Proposal is immutable and can't be changed during the life of the Proposal.
- For courses and programs, where a proposal is associated with a CLU, the objectld references the "shell" CLU.
- Type setup is part of configuration and not managed specifically through the service, including proposalType, proposalReferenceType, referenceObjectType and kewDocumentType.

## (CM 2.0) Proposal Types and States

### Types

#### Proposal Types

Key	Name	Description	Status	In Constant File?	In Database?	Notes/Comments
kuali.proposal.type.course.create	Create Course Proposal	A create course proposal type	Configured in R1			
kuali.proposal.type.course.modify	Modify Course Proposal	A modify course proposal type	Configured in R1			
kuali.proposal.type.course.retire	Retire Course Proposal	A retire course proposal type	Configured in R1			

### Reference Types

Key	Name	Description	Status	In Constant File?	In Database?	Notes/Comments
kuali.proposal.referenceType.clu	Clu Proposal Reference	Clu proposal reference type	Configured in R1			This should be a Reference Object Type Key (refObjectUri)

### KEW Proposal Related Document Types

Key	Name	Description	Status	Java Post Processor	Notes/Questions
kuali.proposal.type.course.retire	Credit Course Retirement	Credit Course Retirement	Configured in R1	org.kuali.student.lum.workflow.CoursePostProcessorBase	These are KS configured RICE types
kuali.proposal.type.course.modify	Credit Course Modification	Credit Course Modification	Configured in R1	org.kuali.student.lum.workflow.CoursePostProcessorBase	
kuali.proposal.type.course.create	Credit Course Proposal	Credit Course Proposal	Configured in R1	org.kuali.student.lum.workflow.CoursePostProcessorBase	
kuali.proposal.type.course.retire	Credit Course Retirement	Credit Course Retirement	Configured in R1	org.kuali.student.lum.workflow.CoursePostProcessorBase	
kuali.proposal.type.course.modify	Credit Course Modification	Credit Course Modification	Configured in R1	org.kuali.student.lum.workflow.CoursePostProcessorBase	
kuali.proposal.type.course.create	Credit Course Proposal	Credit Course Proposal	Configured in R1	org.kuali.student.lum.workflow.CoursePostProcessorBase	
kuali.proposal.type.course.create.admin	Credit Course Admin Proposal	Credit Course Admin Proposal	Configured in R1	org.kuali.student.lum.workflow.CoursePostProcessorBase	
kuali.proposal.type.course.modify.admin	Modify Credit Course Admin Proposal	Modify Credit Course Admin Proposal	Configured in R1	org.kuali.student.lum.workflow.CoursePostProcessorBase	
kuali.proposal.type.majorDiscipline.create	Create Major Discipline Proposal	Create Major Discipline Proposal	Configured in R1	org.kuali.student.lum.workflow.ProgramPostProcessorBase	
kuali.proposal.type.majorDiscipline.modify	Modify Major Discipline Proposal	Modify Major Discipline Proposal	Configured in R1	org.kuali.student.lum.workflow.ProgramPostProcessorBase	

### (CM 2.0) KEW Proposal Related Document Types

## KEW Proposal Related Document Types

Key	Name	Description	Status	Java Post Processor	Notes/Questions
kuali.proposal.type.course.retire	Credit Course Retirement	Credit Course Retirement	Configured in R1	org.kuali.student.lum.workflow.CoursePostProcessorBase	These are KS configured RICE types
kuali.proposal.type.course.modify	Credit Course Modification	Credit Course Modification	Configured in R1	org.kuali.student.lum.workflow.CoursePostProcessorBase	
kuali.proposal.type.course.create	Credit Course Proposal	Credit Course Proposal	Configured in R1	org.kuali.student.lum.workflow.CoursePostProcessorBase	
kuali.proposal.type.course.retire	Credit Course Retirement	Credit Course Retirement	Configured in R1	org.kuali.student.lum.workflow.CoursePostProcessorBase	
kuali.proposal.type.course.modify	Credit Course Modification	Credit Course Modification	Configured in R1	org.kuali.student.lum.workflow.CoursePostProcessorBase	
kuali.proposal.type.course.create	Credit Course Proposal	Credit Course Proposal	Configured in R1	org.kuali.student.lum.workflow.CoursePostProcessorBase	
kuali.proposal.type.course.create.admin	Credit Course Admin Proposal	Credit Course Admin Proposal	Configured in R1	org.kuali.student.lum.workflow.CoursePostProcessorBase	
kuali.proposal.type.course.modify.admin	Modify Credit Course Admin Proposal	Modify Credit Course Admin Proposal	Configured in R1	org.kuali.student.lum.workflow.CoursePostProcessorBase	
kuali.proposal.type.majorDiscipline.create	Create Major Discipline Proposal	Create Major Discipline Proposal	Configured in R1	org.kuali.student.lum.workflow.ProgramPostProcessorBase	
kuali.proposal.type.majorDiscipline.modify	Modify Major Discipline Proposal	Modify Major Discipline Proposal	Configured in R1	org.kuali.student.lum.workflow.ProgramPostProcessorBase	

## (CM 2.0) Proposal Types

### Proposal Types

Key	Name	Description	Status	In Constant File?	In Database?	Notes/Comments
kuali.proposal.type.course.create	Create Course Proposal	A create course proposal type	Configured in R1			
kuali.proposal.type.course.modify	Modify Course Proposal	A modify course proposal type	Configured in R1			
kuali.proposal.type.course.retire	Retire Course Proposal	A retire course proposal type	Configured in R1			

## (CM 2.0) Reference Types

### Reference Types

Key	Name	Description	Status	In Constant File?	In Database?	Notes/Comments

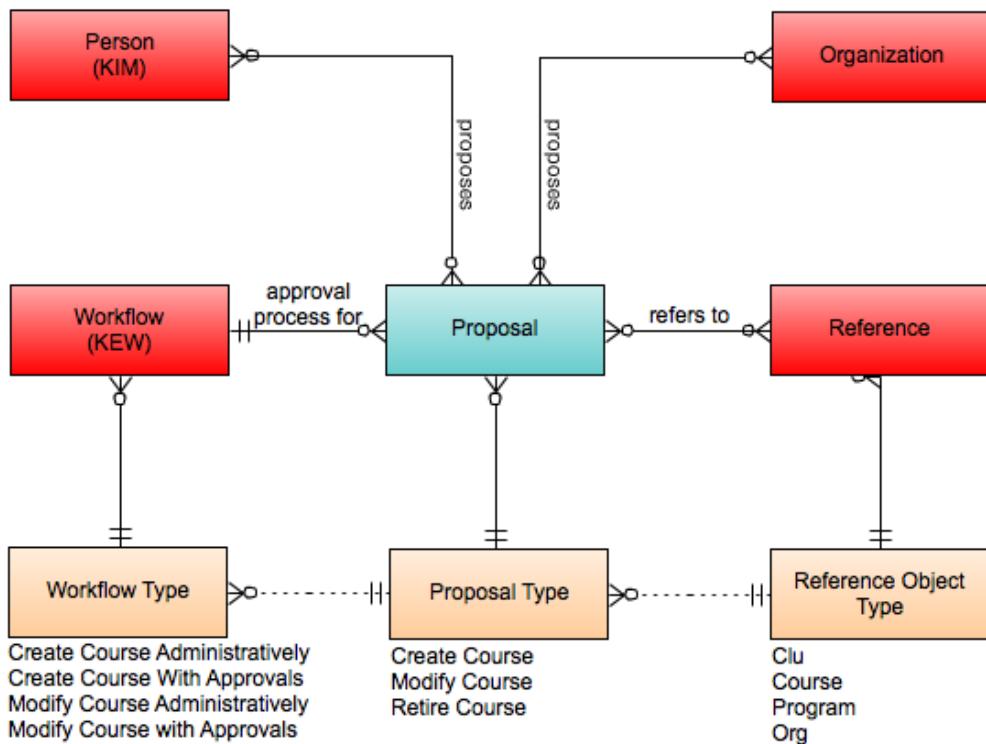
kuali.proposal.referenceType.clu	Clu Proposal Reference	Clu proposal reference type	Configured in R1			This should be a Reference Object Type Key (refObjectUri)
----------------------------------	------------------------	-----------------------------	------------------	--	--	-----------------------------------------------------------

## (CM 2.0) Proposal Service Entity Diagram

- Proposal Service Entity Diagram
- Database Model and DDL

### Proposal Service Entity Diagram

#### Proposal Service Entity Diagram



### Database Model and DDL

- KSEN\_PROPOSAL.pdf
- KSEN\_PROPOSAL\_DDL\_Tables.ddl
- KSEN\_PROPOSAL\_DDL\_Constraints.ddl

## (CM 2.0) Search Service

- Class
- Description
- Entity Diagram
- Service Contract Documentation
- Type and State Configuration
  - LUM Searches

## Class

Search Service is a Core service that supports typed (canned) queries.

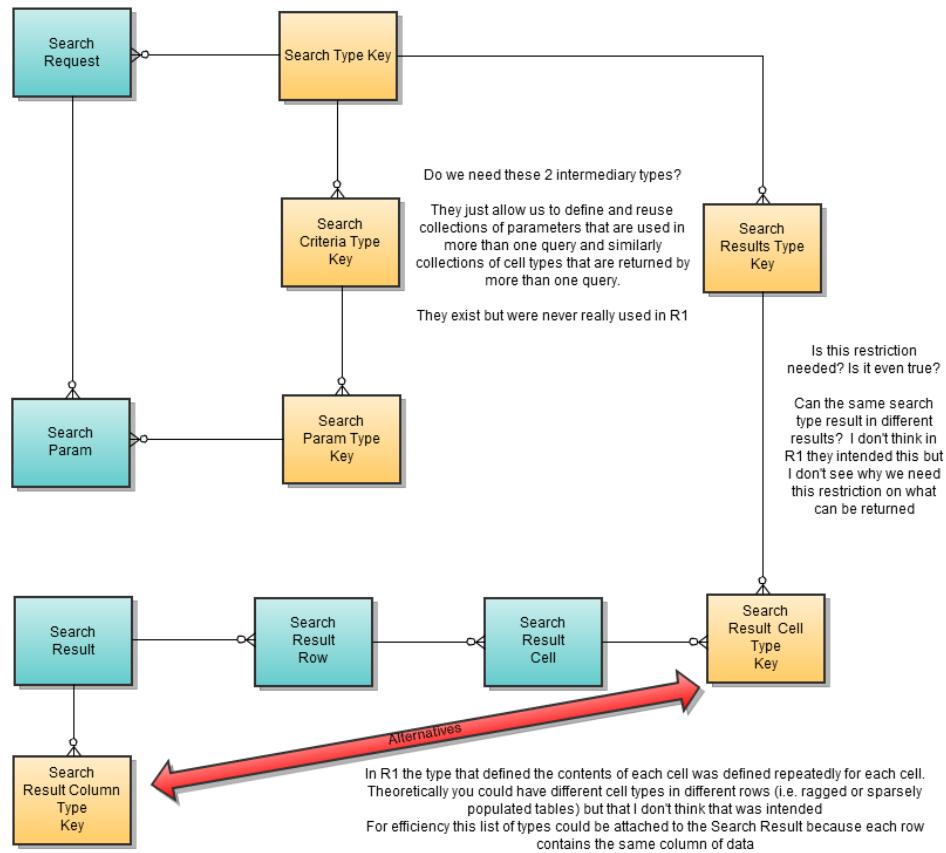
## Description

The Search Service supports the creation of typed searches that query the underlying data store and return results in tabular form. Conceptually it can be viewed as a wrapper for predefined (typed) SQL statements. It has two main purposes:

1. The definitions of searches to meet specific purposes without modifying the contract.
2. The definition of searches that return data other than objects and their attributes, for example counts or sums.

More...

## Entity Diagram



More...

## Service Contract Documentation

[Go to Search Service Contract Documentation](#)

## Type and State Configuration

## LUM Searches

- (CM 2.0) Formatted View of LO Searches *Learning Objectives*
- (CM 2.0) Formatted View of LRC Searches *Learning Results Catalog*
- (CM 2.0) Formatted View of CLU Searches *Canonical Learning Unit*

More...

## (CM 2.0) Search Service Description and Assumptions

### Search Service

#### Description

The Search Service supports the creation of typed searches that query the underlying data store and return results in tabular form. Conceptually it can be viewed as a wrapper for predefined (typed) SQL statements. It has two main purposes:

1. The definitions of searches to meet specific purposes without modifying the contract.
2. The definition of searches that return data other than objects and their attributes, for example counts or sums.

#### Key Concepts

##### searchType

- Equivalent to a named search or parameterized query. This describes the inputs as well as the outputs from calling a search of this type. Since the criteria and results are uniquely named, this can be seen also as the combination of those two concepts.
- Search types provide a handle to both the criteria to be provided and the results to be returned.
- A user may be authorized to be able to perform a type of search even if they are not authorized in the general case to view the more complete information of the entities involved. This provides something akin to view functionality, albeit in a constrained fashion as the results have been flattened.
- The search results returned may also be dependent on underlying authorization. In other words, certain searches may limit the returned values by the granted authorizations.

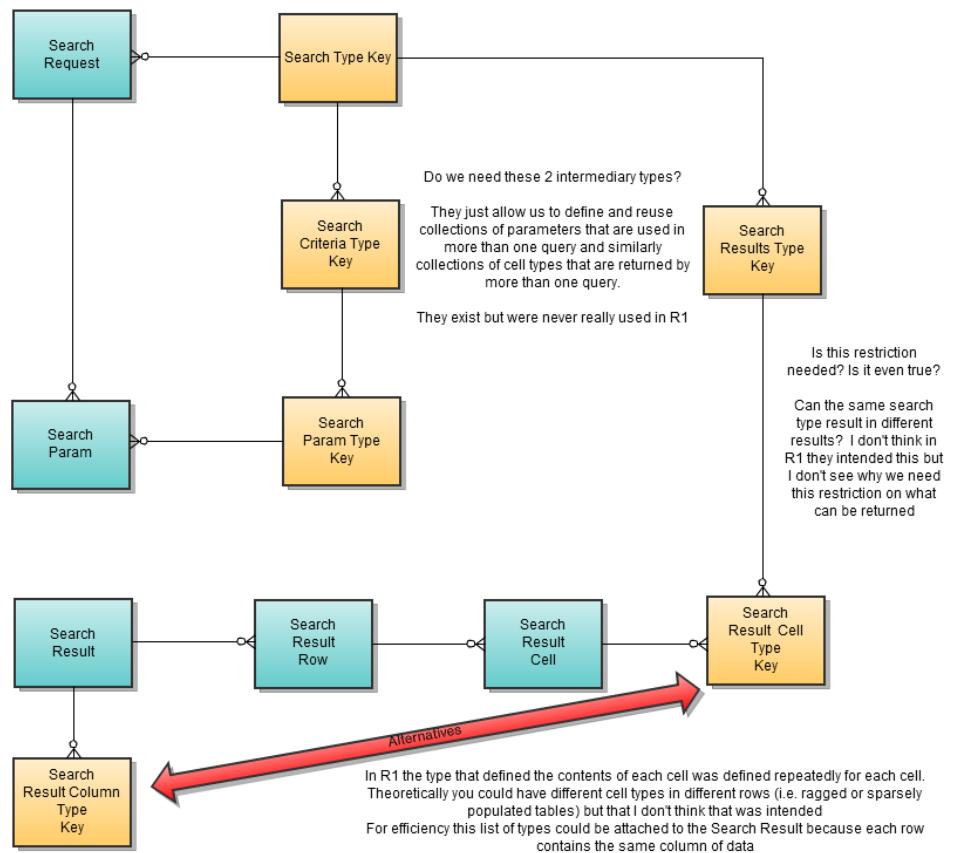
#### Assumptions

- Type setup is part of configuration and not managed specifically through the service
- The query itself is unlikely to be directly exposed.
- Meta data about the allowed search parameters and expected results is deferred and is not part of this contract but will be considered in future versions.
  - searchResultType
    - Describes the set of columns returned in a search. This concept is called out to allow building standard representations in an application around particular sets. This would allow for some customization at the application level regarding ordering, display, or even secondary usage of values in a particular component.
    - For example, multiple searches may be configured which return the official identifier and short name of a course, and so the application can treat this set of information as it would a component.
  - resultColumn
    - Describes the contents of a column of results returned by the search, including concepts such as the data type (string, date, number, etc.).
  - searchCriteriaType
    - Describes a set of parameters to a search. This concept is called out to allow building standard representations in an application around particular sets. This would allow for some customization at the application level regarding ordering, display, or even defaulting of values in a particular component.
    - For example, multiple searches may be configured which take solely a first and last name for a person as criteria, which would allow that bit to be known and rendered appropriately.
  - queryParam
    - Describes a query parameter, including concepts such as data type, allowed values, length/character constraints, and required/optional status.

## (CM 2.0) Search Service Entity Diagram

- Search Service Entity Diagram
- Database Entity Model and DDL

### Search Service Entity Diagram



## Database Entity Model and DDL

Search has no persistence so there is no underlying database model.

## (CM 2.0) Search Types and States

- Formatted versions of search config dictionaries
  - LUM Searches
  - Core Searches
  - Hardcoded searches

### Formatted versions of search config dictionaries

#### LUM Searches

- (CM 2.0) Formatted View of LO Searches *Learning Objectives*
- (CM 2.0) Formatted View of LRC Searches *Learning Results Catalog*
- (CM 2.0) Formatted View of CLU Searches *Canonical Learning Unit*

#### Core Searches

- (CM 2.0) Formatted View of ATP Searches *Academic Time Period*
- (CM 2.0) Formatted View of Comment Searches
- (CM 2.0) Formatted View of EM Searches *Enumeration Management*
- (CM 2.0) Formatted View of Organization Searches
- (CM 2.0) Formatted View of Proposal Searches
- (CM 2.0) Formatted View of Statement Searches
- (CM 2.0) Formatted View of Subject Code Searches *Subject Area*

#### Hardcoded searches

This documents the hardcoded searches. These are searches that do not use the XML files to generate JPQL but rather use java code making queries and calls to piece together the information. Often this was done for efficiency reasons.

 To override these you need to override the java program not the search-config.xml.

Project	Java Program	Type	Notes
ks-common-impl	SearchableCrudDaolmpl	XML Types	This is the impl for the XML defined searches
ks-common-ui	SearchDispatchRpcGwtServlet		does some sort of translation of version id lu.queryParam.cluVersionIndId
ks-core-impl	OrganizationServiceImpl		Calles hard wired searchOperations first then calls XML configured searches next
ks-core-impl	OrganizationHierarchySearch	org.search.orgQuickViewByRelationTypeOrgTypeRelatedOrgIds	The type is defined when it is injected into the searchOperations of the OrganizationServiceImpl See ks-core-no-tx-context.xml
ks-core-impl	PersonSearchServiceImpl		Dispatcher configured with just the QuickViewByGivenName hard wired search. See ks-core-no-tx-context.xml
ks-core-impl	QuickViewByGivenName	person.search.personQuickView ByGivenName	Hardwired to call KIM Identity Service
ks-core-impl	ProposalServiceImpl	proposal.search.proposalsForReferencelids	Intercepts type and called doSearchProposalsForReferencelids which is private, should be protected
ks-core-impl	StatementServiceImpl	stmt.search.dependencyAnalysis	Also does some funky stuff with the parameters.
ks-core-impl	SubjectCodeServiceImpl	subjectCode.search.subjectCodeGeneric	has private method doSubjectCodeGenericSearch should be protected
ks-core-impl	SubjectCodeServiceImpl	subjectCode.search.orgsForSubjectCode	private method doOrgsForSubjectCodeSearch should be protected
ks-core-impl	EnumerationManagementServiceImpl	enumeration.management.search	BUG?... this one search is hardwired but then it does not shell out to the XML impls and instead returns null
ks-enroll-impl	CourseOfferingHistorySearchImpl	kuali.search.type.lui.pastCourse Offering	Hard wired search to get past course offerings

ks-lum-impl	LearningObjectiveServiceImpl	lo.search.loByCategory	This is actually an XML defined one but this does a hardwired grouping by category if that ordering is requested
ks-lum-impl	LRCServiceImpl	lrc.search.resultValue	Hard wired search then shells off to XML searches
ks-lum-impl	CluServiceImpl	lu.search.dependencyAnalysis	doDependencyAnalysisSearch
ks-lum-impl	CluServiceImpl	lu.search.browseProgram	doBrowseProgramSearch
ks-lum-impl	CluServiceImpl	lu.search.browseVariations	doBrowseProgramSearch
ks-lum-impl	CluServiceImpl	lrc.search.resultComponent	doResultComponentTypesForCluSearch
ks-lum-impl	CluServiceImpl	lu.search.proposalsByCourseCode	doSearchProposalsByCourseCode
ks-lum-impl	CluServiceImpl	lu.search.clu.versions	doBrowseVersionsSearch
ks-lum-impl	CluServiceImpl	lu.search.resultComponents	doResultComponentTypesForCluSearch
ks-lum-impl	CluServiceImpl	cluset.search.generic	fixes up some paramter
ks-lum-impl	CluServiceImpl	cluset.search.genericWithClus	as part of previous fixup
ks-lum-impl	TypeSearch		generic search framework for types configured above
ks-lum-impl	TypeSearchServiceImpl		generic search framework for types configured above
ks-lum-impl	AtpDurationTypeSearch	atp.search.atpDurationTypes	Wired into the TypeSearchServiceImpl see ks-lu-no-context.xml
ks-lum-impl	AtpSeasonTypeSearch	atp.search.atpSeasonTypes	Wired into the TypeSearchServiceImpl see ks-lu-no-context.xml
ks-lum-impl	ResultComponentTypeSearch	lrc.search.resultComponentType	Wired into the TypeSearchServiceImpl see ks-lu-no-context.xml
ks-common-impl	ApptWindowCountsSearchImpl	kuali.search.type.appt.appointmentCountsForWindowId	Really an enrollment search
ks-common-impl	SearchServiceHardwiredImpl	appt.search.appointmentCountForWindowId	Really same as previous one!

## (CM 2.0) Formatted View of ATP Searches

⚠ This page was automatically generated on Tue Dec 11 12:23:31 EST 2012  
DO NOT UPDATE MANUALLY!

This page represents a formatted view of [atp-search-config.xml](#)

- All ATP types (atp.search.atpTypes)
  - 0 Possible Search Criteria
  - 2 Result Columns Returned
  - JPQL Implementation
- ShortName match (atp.search.atpByShortName)
  - 1 Possible Search Criteria

- 2 Result Columns Returned
  - JPQL Implementation
  - Advanced ATP match (atp.search.advancedAtpSearch)
    - 11 Possible Search Criteria
    - 5 Result Columns Returned
    - JPQL Implementation
  - Date match (atp.search.atpByDate)
    - 1 Possible Search Criteria
    - 2 Result Columns Returned
    - JPQL Implementation
- 

## All ATP types (atp.search.atpTypes)

<b>Search Type Key</b>	atp.search.atpTypes
<b>Name</b>	All ATP types
<b>Description</b>	Returns the list of all ATP Types

## 0 Possible Search Criteria

<b>Type Key</b>	atp.criteria.atpType
<b>Name</b>	AtpTypesCriteria
<b>Description</b>	Criteria for searching of a list of all ATP Type ids

Name	Optional	DataType	Read Only	Type Key	Implementation
------	----------	----------	-----------	----------	----------------

## 2 Result Columns Returned

<b>Type Key</b>	atp.result.atpType
<b>Name</b>	ATP Types
<b>Description</b>	ATP Types

Name	DataType	Type Key
ATP Type Identifier	string	atp.resultColumn.atpTypeId
Type Name	string	atp.resultColumn.atpTypeName

## JPQL Implementation

```
SELECT atptype.id, atptype.name FROM AtpType atptype
```

## ShortName match (atp.search.atpByShortName)

<b>Search Type Key</b>	atp.search.atpByShortName
<b>Name</b>	ShortName match
<b>Description</b>	Search on shortName only.

## 1 Possible Search Criteria

<b>Type Key</b>	atp.criteria.atpByShortName
<b>Name</b>	AtpByShortNameCriteria

<b>Description</b>			AtpByShortNameCriteria Description		
Name	Optional	DataType	Read Only	Type Key	Implementation
Name	false	string	true	atp.queryParam.atpShortName	null

## 2 Result Columns Returned

Type Key	atp.result.atpQuickView	
Name	ATP Quick View	
Description	Quick view of the ATP	

Name	DataType	Type Key
ATP Identifier	string	atp.resultColumn.atpId
ATP Short Name	string	atp.resultColumn.atpShortName

## JPQL Implementation

```
SELECT atp.id, atp.name FROM Atp atp WHERE atp.name
    like :atp_queryParam_atpShortName or atp.id =
    :atp_queryParam_atpId
```

## Advanced ATP match (atp.search.advancedAtpSearch)

Search Type Key	atp.search.advancedAtpSearch
Name	Advanced ATP match
Description	Search by Name, start date, end date, and type

## 11 Possible Search Criteria

Type Key	atp.criteria.advancedAtpSearch
Name	AdvancedAtpSearchCriteria
Description	

### AdvancedAtpSearchCriteria Description

Name	Optional	DataType	Read Only	Type Key	Implementation
ATP Short Name	true	string	true	atp.advancedAtpSearchParam.atpShortName	atp.name

ATP ID	true	string	true	atp.advancedAtpSearchParam.atpId	atp.i d = :atp_ advan cedAt pSear chPar am_at pId
Start Date	true	date	true	atp.advancedAtpSearchParam.atpStartDate	atp.s tartD ate = :atp_ advan cedAt pSear chPar am_at pStar tDate
End Date	true	date	true	atp.advancedAtpSearchParam.atpEndDate	atp.e ndDat e = :atp_ advan cedAt pSear chPar am_at pEndD ate
Type Name	true	string	true	atp.advancedAtpSearchParam.atpTypeName	atp.t ype.n ame

Type	true	null	false	atp.advancedAtpSearchParam.atpType	atp.a tpTyp e IN (:atp _adva ncedA tpSea rchPa ram_a tpTyp e)
ATP IDe	true	string	true	atp.advancedAtpSearchParam.atpStartDateAtpConstraintId	((:atp_adv anced AtpSe archP aram_ atpSt artDa teAtp Const raint Id IS NULL OR (atp. start Date >= (SELE CT MAX(s tartA tp.st artDa te) FROM Atp start Atp WHERE start Atp.i d = :atp_ advan cedAt pSear chPar am_at pStar tDate AtpCo

```
nstra
intId
)))
AND

(:atp
_adva
ncedA
tpSea
rchPa
ram_a
tpSta
rtDat
eAtuC
onstr
aintI
d IS
NULL
OR
(atp.
endDa
te >=
(SELE
CT
MAX(s
tartA
tp.en
dDate
)
FROM
Atp
start
Atp
WHERE
start
Atp.i
d =
:atp_
advan
cedAt
pSear
chPar
am_at
pStar
tDate
AtpCo
```

					nstra intId ))))
ATP IDe	true	string	true	atp.advancedAtpSea rchParam.atpEndDa teAtpConstraintId	( :atp _adva ncedA tpSea rchPa ram_a tpEnd DateA tpCon strai ntId IS NULL OR (atp. endDa te <= (SELE CT MAX(e ndAtp .endD ate) FROM Atp endAt p WHERE endAt p.id = :atp_ advan cedAt pSear chPar am_at pEndD ateAt pCons train tId)) )
ATP IDe	true	string	true	atp.advancedAtpSea rchParam.atpStartD ateAtpConstraintIdE xclusive	((:at p_adv anced AtpSe archP

```
aram_
atpSt
artDa
teAtp
Const
raint
IdExc
lusiv
e IS
NULL
OR
(atp.
start
Date
>
(SELE
CT
MAX(s
tartA
tp.st
artDa
te)
FROM
Atp
start
Atp
WHERE
start
Atp.i
d =
:atp_
advan
cedAt
pSear
chPar
am_at
pStar
tDate
AtpCo
nstra
intId
Exclu
sive)
))
AND

(:atp
_adva
ncedA
tpSea
rchPa
ram_a
tpSta
rtDat
eAtpC
onstr
aintI
dExcl
usive
```

IS  
NULL  
OR  
(atp.  
endDa  
te >  
(SELE  
CT  
MAX(s  
tartA  
tp.en  
dDate  
)  
FROM  
Atp  
start  
Atp  
WHERE  
start  
Atp.i  
d =  
:atp\_  
advan  
cedAt  
pSear  
chPar  
am\_at  
pStar  
tDate  
AtpCo  
nstra  
intId

					Exclusive))
ATP IDe	true	string	true	atp.advancedAtpSearchParam.atpEndDateAtpConstraintIdExclusive	(:atp_advanc edAtpSea rchPar am_a tpEnd DateA tpCon strai ntIdE xclusive IS NULL OR (atp.endDa te < (SELECT MAX(endAtp.endD ate) FROM Atp endAt p WHERE endAt p.id = :atp_advanc edAt pSear chPar am_at pEndD ateAt pCons train tIdEx clusive)))

ATP IDe	true	string	true	atp.advancedAtpSearchParam.optionalAtplds	atp.id IN (:atp_advancdA_tpSearchParam_optionalAtpIds)
---------	------	--------	------	-------------------------------------------	--------------------------------------------------------

## 5 Result Columns Returned

Type Key	atp.result.advancedAtpSearch
Name	ATP Advanced Search
Description	

Search by Name, start date, end date, and type

Name	DataType	Type Key
ATP Identifier	string	atp.resultColumn.atpId
ATP Short Name	string	atp.resultColumn.atpShortName
Start Date	date	atp.resultColumn.atpStartDate
Type Name	string	atp.resultColumn.atpTypeName
Plain Description	string	atp.resultColumn.atpDescrPlain

## JPQL Implementation

```

SELECT
    atp.id,
    atp.name,
    atp.startDate,
    atp.atpType,
    atp.plain
FROM AtpEntity atp

```

## Date match (atp.search.atpByDate)

Search Type Key	atp.search.atpByDate
Name	Date match
Description	Search on shortName only.

## 1 Possible Search Criteria

Type Key	atp.criteria.atpByDate
----------	------------------------

<b>Name</b>	Atp by Date criteria				
<b>Description</b>	Atp by Date criteria				
Name	Optional	DataType	Read Only	Type Key	Implementation
Date	false	date	false	atp.queryParam.searchDate	null

## 2 Result Columns Returned

<b>Type Key</b>	atp.result.atpByDate	
<b>Name</b>	ATP By Date	
<b>Description</b>	ATP By Date	

Name	DataType	Type Key
ATP Identifier	string	atp.resultColumn.atpld
Do you mean	string	atp.resultColumn.atpByDateDisplay

## JPQL Implementation

```

SELECT
    atp.id,
    concat(concat(concat(seasonaltype.name, ' '),
    durtype.name), ' ', ' ),
    atptype.name)
        FROM Atp atp
    JOIN atp.type atptype
    JOIN atptype.seasonalType seasonaltype
    JOIN atptype.durationType durtype
    WHERE
    :atp_queryParam_searchDate BETWEEN atp.startDate AND atp.endDate

```

## (CM 2.0) Formatted View of CLU Searches

⚠ This page was automatically generated on Tue Dec 11 12:23:37 EST 2012  
DO NOT UPDATE MANUALLY!

This page represents a formatted view of lu-search-config.xml

- Search for most recent LUs (lu.search.mostCurrent.union)
  - 14 Possible Search Criteria
  - 12 Result Columns Returned
  - JPQL Implementation
- Search for resultComponents for a specific Clu (lu.search.resultComponents)
  - 3 Possible Search Criteria
  - 1 Result Columns Returned
  - JPQL Implementation
- Determine if the course version is versionable (lu.search.isVersionable)
  - 3 Possible Search Criteria
  - 1 Result Columns Returned
  - JPQL Implementation
- Proposal search by course codes (lu.search.proposalsByCourseCode)
  - 1 Possible Search Criteria
  - 3 Result Columns Returned
  - JPQL Implementation
- Quick search for current courses (lu.search.current.quick)

- 5 Possible Search Criteria
  - 3 Result Columns Returned
  - JPQL Implementation
- Browse Program (lu.search.browseProgram)
  - 0 Possible Search Criteria
  - 9 Result Columns Returned
  - JPQL Implementation
- Browse Variations (lu.search.browseVariations)
  - 0 Possible Search Criteria
  - 2 Result Columns Returned
  - JPQL Implementation
- Dependency Analysis (lu.search.dependencyAnalysis)
  - 1 Possible Search Criteria
  - 12 Result Columns Returned
  - JPQL Implementation
- Basic and Advanced Search (lu.search.current)
  - 14 Possible Search Criteria
  - 12 Result Columns Returned
  - JPQL Implementation
- Basic and Advanced Search (lu.search.noncurrent)
  - 14 Possible Search Criteria
  - 12 Result Columns Returned
  - JPQL Implementation
- Basic and Advanced Search (lu.search.generic)
  - 14 Possible Search Criteria
  - 12 Result Columns Returned
  - JPQL Implementation
- Lum Search for Clus (lu.search.clus)
  - 0 Possible Search Criteria
  - 3 Result Columns Returned
  - JPQL Implementation
- Lum Search for Clus in a Cluset (lu.search.clusInCluset)
  - 1 Possible Search Criteria
  - 4 Result Columns Returned
  - JPQL Implementation
- Basic and Advanced Search (lu.search.cluCluRelation)
  - 1 Possible Search Criteria
  - 2 Result Columns Returned
  - JPQL Implementation
- Lum Search for Clus Ids and Codes (lu.search.allLoQuickView)
  - 8 Possible Search Criteria
  - 6 Result Columns Returned
  - JPQL Implementation
- Search for clu id by its code (lu.search.cluByCode)
  - 0 Possible Search Criteria
  - 1 Result Columns Returned
  - JPQL Implementation
- Code and state match (lu.search.cluByCodeAndState)
  - 2 Possible Search Criteria
  - 3 Result Columns Returned
  - JPQL Implementation
- Basic and Advanced Search on Clu Sets (cluset.search.generic)
  - 11 Possible Search Criteria
  - 4 Result Columns Returned
  - JPQL Implementation
- Basic and Advanced Search on Clu Sets (cluset.search.genericWithClus)
  - 11 Possible Search Criteria
  - 4 Result Columns Returned
  - JPQL Implementation
- Search for clu type optionally by code (lu.search.all.lu.Types)
  - 2 Possible Search Criteria
  - 5 Result Columns Returned
  - JPQL Implementation
- Find all versions of a clu (lu.search.clu.versions)
  - 1 Possible Search Criteria
  - 16 Result Columns Returned
  - JPQL Implementation
- Basic Search forPublication Types (lu.search.publication.types)
  - 2 Possible Search Criteria
  - 2 Result Columns Returned
  - JPQL Implementation
- LU Search for Majors and Variations (lu.search.union.majors)

- 6 Possible Search Criteria
  - 8 Result Columns Returned
  - JPQL Implementation
  - Search for Majors (lu.search.luProgramsByName)
    - 6 Possible Search Criteria
    - 8 Result Columns Returned
    - JPQL Implementation
  - Search for Majors (lu.search.luVariationsByName)
    - 6 Possible Search Criteria
    - 8 Result Columns Returned
    - JPQL Implementation
  - Find clu's based on the admin org (lu.search.by.admin.org)
    - 15 Possible Search Criteria
    - 14 Result Columns Returned
    - JPQL Implementation
  - Search for Majors (lu.search.luByRelation)
    - 2 Possible Search Criteria
    - 13 Result Columns Returned
    - JPQL Implementation
- 

### Search for most recent LUs (lu.search.mostCurrent.union)

<b>Search Type Key</b>	lu.search.mostCurrent.union
<b>Name</b>	Search for most recent LUs
<b>Description</b>	LU Search that finds all current courses or most recent approved courses for non-active LUs

### 14 Possible Search Criteria

<b>Type Key</b>	lu.criteria.luAdvancedCriteria
<b>Name</b>	LuAdvancedCriteria
<b>Description</b>	Advanced criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
CLU Identifier	true	string	true	lu.queryParam.luOptionalId	clu.id = :lu_queryParam_luOptionalId
Short Name	true	string	false	lu.queryParam.luOptionalShortName	clu.optionalIdentifier.shortName

Long Name	true	string	false	luQueryParam.luOptionalLongName	clu.officeId.alide ntifier.longName
Learning Unit Type	true	string	false	luQueryParam.luOptionalType	clu.uType.id IN (:lu_queryParam_luOptionalType)
Code	true	string	false	luQueryParam.luOptionalCode	LOWER(clu.officeId.alide ntifier.code) LIKE '%'    LOWER(:lu_queryParam_luOptionalCode)    '%'
Division	true	string	false	luQueryParam.luOptionalDivision	clu.officeId.alide ntifier.division

Level	true	string	false	luQueryParam.luOptionalLevel	clu.offici.alIden.tifi.er.level
Clu Description	true	string	false	luQueryParam.luOptionalDescr	clu.descri plain
Study Subject Area	true	string	false	luQueryParam.luOptionalStudySubjectArea	clu.offici.alIden.tifi.er.di vision
Learning Unit State	true	string	false	luQueryParam.luOptionalState	clu.state IN (:lu_queryParam_luOptionalState)
Suffix Code	true	string	false	luQueryParam.luOptionalCrsNoRange	! !NUM BER_R ANGE clu.offici.alIden.tifi.er.suffixCode

Effective Date	true	date	false	lu.queryParam.luOptionalEffectiveDate1	clue ffect iveDa te >= :lu_q ueryP aram_ luOpt ional Effec tiveD ate1
Effective Date	true	date	false	lu.queryParam.luOptionalEffectiveDate2	clue ffect iveDa te <= :lu_q ueryP aram_ luOpt ional Effec tiveD ate2

Version Independent Id	true	string	false	lu.queryParam.luOptionalVersionIndId	<pre> clu.v ersio n.ver sionI ndId = :lu_q ueryP aram_ luOpt ional Versi onInd Id  AND (clu. versi on.cu rrent Versi onSta rt &lt;= CURRE NT_TI MESTA MP AND (clu. versi on.cu rrent Versi onEnd &gt; CURRE NT_TI MESTA MP OR clu.v ersio n.cur rentV ersio nEnd IS NULL) ) </pre>
------------------------	------	--------	-------	--------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 12 Result Columns Returned

Type Key	lu.result.generic
Name	Lu Full View
Description	Full view of clus

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluId
Name	string	lu.resultColumn.luOptionalShortName
Name	string	lu.resultColumn.luOptionalLongName
Code	string	lu.resultColumn.luOptionalCode
Level	string	lu.resultColumn.luOptionalLevel
Description	string	lu.resultColumn.luOptionalDescr
Study Subject Area	string	lu.resultColumn.luOptionalStudySubjectArea
State	string	lu.resultColumn.luOptionalState
VersionIndId	string	lu.resultColumn.luOptionalVersionIndId
VersionedFromId	string	lu.resultColumn.luOptionalVersionedFromId
VersionStartDate	date	lu.resultColumn.luOptionalVersionStartDate
VersionEndDate	date	lu.resultColumn.luOptionalVersionEndDate

#### JPQL Implementation

```
null
```

#### Search for resultComponents for a specific Clu (lu.search.resultComponents)

Search Type Key	lu.search.resultComponents
Name	Search for resultComponents for a specific Clu
Description	Search for resultComponents for a specific Clu

#### 3 Possible Search Criteria

Type Key	lu.criteria.resultComponents				
Name	ResultComponent Criteria				
Description	ResultComponent Criteria				
Name	Optional	DataType	Read Only	Type Key	Implementation

CLU Identifier	true	string	true	lu.queryParam.resultComponentOptionalType	result.cluResultType.id = :lu_queryParam_resultComponentOptionalType
CLU Identifier	true	string	true	lu.queryParam.luOptionalId	clu.id = :lu_queryParam_luOptionalId
CLU Identifier	true	string	true	lu.queryParam.resultComponentOptionalId	options.resultComponentOptionalId = :lu_queryParam_resultComponentOptionalId

#### 1 Result Columns Returned

Type Key	lu.result.resultComponents	
Name	ResultComponent Results	
Description	ResultComponent Results	

Name	DataType	Type Key
Result Component Key	string	lu.resultColumn.resultComponentId

## JPQL Implementation

```
SELECT DISTINCT options.resultComponentId FROM CluResult result JOIN result.clu clu,
IN(result.resultOptions) options
```

### Determine if the course version is versionable (lu.search.isVersionable)

<b>Search Type Key</b>	lu.search.isVersionable
<b>Name</b>	Determine if the course version is versionable
<b>Description</b>	Determine if the course version is versionable

### 3 Possible Search Criteria

<b>Type Key</b>	lu.criteria.isVersionable
<b>Name</b>	IsVersionable Criteria
<b>Description</b>	IsVersionable Criteria

<b>Name</b>	<b>Optional</b>	<b>DataType</b>	<b>Read Only</b>	<b>Type Key</b>	<b>Implementation</b>
Version Independent Id	false	string	false	lu.queryParam.versionIndId	null
Sequence Number	false	long	false	lu.queryParam.sequenceNumber	null
Learning Unit State	true	string	false	lu.queryParam.luOptionalState	clu.state IN (:lu_queryParam_luOptionalState)

### 1 Result Columns Returned

<b>Type Key</b>	lu.result.isVersionable
<b>Name</b>	Is Versionable Results
<b>Description</b>	Is Versionable Results

<b>Name</b>	<b>DataType</b>	<b>Type Key</b>
IsVersionable	boolean	lu.resultColumn.isVersionable

## JPQL Implementation

```

SELECT COUNT(clu) FROM Clu clu WHERE clu.version.versionIndId =
:lu_queryParam_versionIndId AND clu.version.sequenceNumber >
:lu_queryParam_sequenceNumber

```

### Proposal search by course codes (lu.search.proposalsByCourseCode)

<b>Search Type Key</b>	lu.search.proposalsByCourseCode
<b>Name</b>	Proposal search by course codes
<b>Description</b>	Returns all proposals that contain the course codes

#### 1 Possible Search Criteria

<b>Type Key</b>	lu.criteria.proposalsForReferencelds
<b>Name</b>	Proposal search by course codes Criteria
<b>Description</b>	Proposal search by course codes Criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
Code	true	string	false	lu.queryParam.luOptionalCode	<pre> LOWER (clu. offic ialId entif ier.c ode) LIKE '%'    LOWER (:lu_ query Param _luOp tiona lCode )    '%'</pre>

#### 3 Result Columns Returned

<b>Type Key</b>	lu.result.proposalsForReferencelds
<b>Name</b>	Browse Program Results
<b>Description</b>	View of Program Clus

Name	DataType	Type Key
Proposal Identifier	string	lu.resultColumn.proposalId

Proposal Name	string	lu.resultColumn.proposalOptionalName
Proposal Reference Id	string	lu.resultColumn.proposalOptionalReferenceId

#### JPQL Implementation

```
null
```

#### Quick search for current courses (lu.search.current.quick)

Search Type Key	lu.search.current.quick
Name	Quick search for current courses
Description	Returns all current courses

#### 5 Possible Search Criteria

Type Key	lu.criteria.currentquick
Name	Quick search for current courses
Description	Quick search for current courses

Name	Optional	DataType	Read Only	Type Key	Implementation
CLU Identifier	true	string	true	lu.queryParam.luOptionalId	<pre>clu.i d = :lu_q ueryP aram_ luOpt ional Id</pre>

Code	true	string	false	luQueryParam.luOptionalCode	<pre> LOWER (clu. offic ialId entif ier.c ode) LIKE '%'    LOWER (:lu_ query Param _luOp tiona lCode )    '%' </pre>
Learning Unit Type	true	string	false	luQueryParam.luOptionalType	<pre> clu.l uType .id IN (:lu_ query Param _luOp tiona lType )</pre>
Learning Unit State	true	string	false	luQueryParam.luOptionalState	<pre> clu.s tate IN (:lu_ query Param _luOp tiona lStat e)</pre>

Version Independent Id	true	string	false	lu.queryParam.luOptionalVersionIndId	<pre> clu.v ersio n.ver sionI ndId = :lu_q ueryP aram_ luOpt ional Versi onInd Id  AND (clu. versi on.cu rrent Versi onSta rt &lt;= CURRE NT_TI MESTA MP AND (clu. versi on.cu rrent Versi onEnd &gt; CURRE NT_TI MESTA MP OR clu.v ersio n.cur rentV ersio nEnd IS NULL) ) </pre>
------------------------	------	--------	-------	--------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### 3 Result Columns Returned

Type Key	lu.result.currentquick
Name	Quick search for current courses
Description	View of Program Clus

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluId
Code	string	lu.resultColumn.luOptionalCode
VersionIndId	string	lu.resultColumn.luOptionalVersionIndId

### JPQL Implementation

```

SELECT clu.id, officialIdentifier.code,
clu.version.versionIndId
    FROM Clu clu
    LEFT JOIN clu.officialIdentifier officialIdentifier
      WHERE clu.version.sequenceNumber=(SELECT
MAX(mostrecentclu.version.sequenceNumber)
    FROM Clu mostrecentclu
    WHERE
clu.version.versionIndId=mostrecentclu.version.versionIndId
      AND
mostrecentclu.version.versionIndId IS NOT NULL
    GROUP BY
mostrecentclu.version.versionIndId)

```

### Browse Program (lu.search/browseProgram)

Search Type Key	lu.search/browseProgram
Name	Browse Program
Description	Search for all Program Clus

### 0 Possible Search Criteria

Type Key	lu.criteria/browseProgram
Name	Browse Program Criteria
Description	Browse Program Criteria

Name	Optional	DataType	Read Only	Type Key	Implementation

### 9 Result Columns Returned

Type Key	lu.result/browseProgram
Name	Browse Program Results
Description	View of Program Clus

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluId
Name	string	lu.resultColumn.luOptionalLongName
State	string	lu.resultColumn.luOptionalState

Result Component Key	string	lu.resultColumn.resultComponentId
Department	string	lu.resultColumn.luOptionalAdminOrg
Name	string	lu.resultColumn.variationId
Campus Location	string	lu.resultColumn.luOptionalCampusLocation
Level	string	lu.resultColumn.cluOfficialIdentifier.level
Clu Id	string	lu.resultColumn.credentialCluld

### JPQL Implementation

```

SELECT clu.id, ident.longName, clu.state, options.resultComponentId, org.orgId, '',
campus.campusLocation, credentialIdent.level, rel.clu.id
    FROM CluIdentifier ident, CluResult result,
IN(result.resultOptions) options, Clu clu, IN(clu.adminOrgs) org,
IN(clu.campusLocations) campus, CluCluRelation rel, CluIdentifier credentialIdent
        WHERE clu.officialIdentifier.id = ident.id
        AND clu.luType.id = 'kuali.lu.type.MajorDiscipline'
        AND result.clu.id = clu.id
        AND result.cluResultType.id = 'kuali.resultType.degree'
        AND org.type = 'kuali.adminOrg.type.CurriculumOversightDivision'
        AND clu.state IN ('Active','Suspended','Retired')
        AND rel.relatedClu.id = clu.id
        AND rel.luLuRelationType.id =
'kuali.lu.lu.relation.type.hasMajorProgram'
            AND rel.clu.officialIdentifier.id = credentialIdent.id

```

### Browse Variations (lu.search.browseVariations)

Search Type Key	lu.search.browseVariations
Name	Browse Variations
Description	Search for all Variation Clus

### 0 Possible Search Criteria

Type Key	lu.criteria/browseVariations
Name	Browse Variations Criteria
Description	Browse Variations Criteria

Name	Optional	DataType	Read Only	Type Key	Implementation

### 2 Result Columns Returned

Type Key	lu.result.browseVariations
Name	Browse Program Results
Description	View of Program Clus

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluld

Name	string	lu.resultColumn.luOptionalLongName
------	--------	------------------------------------

### JPQL Implementation

```

SELECT rel.clu.id, rel.relatedClu.officialIdentifier.longName
      FROM CluCluRelation rel
      WHERE rel.luLuRelationType.id =
'kuali.lu.lu.relation.type.hasVariationProgram'
      AND rel.relatedClu.luType.id = 'kuali.lu.type.Variation' ORDER BY
rel.relatedClu.officialIdentifier.longName ASC

```

### Dependency Analysis (lu.search.dependencyAnalysis)

Search Type Key	lu.search.dependencyAnalysis
Name	Dependency Analysis
Description	Search for Clus that are dependencies of the target Clu

### 1 Possible Search Criteria

Type Key	lu.criteria.dependencyAnalysis
Name	Dependency Analysis Criteria
Description	Dependency Analysis Criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
CLU Identifier	true	string	true	lu.queryParam.luOptionalCluId	null

### 12 Result Columns Returned

Type Key	lu.result.dependencyAnalysis
Name	Dependency Analysis Results
Description	Full view of clus/clusets that are dependencies of the target Clu

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluId
Clu Type	string	lu.resultColumn.cluType
Name	string	lu.resultColumn.luOptionalShortName
Name	string	lu.resultColumn.luOptionalLongName
Code	string	lu.resultColumn.luOptionalCode
Dependency Type	string	lu.resultColumn.luOptionalDependencyType
Dependency Type Name	string	lu.resultColumn.luOptionalDependencyTypeName
Dependency Root Id	string	lu.resultColumn.luOptionalDependencyRootId

Dependency Requirement Component Ids	string	lu.resultColumn.luOptionalDependencyRequirementComponentIds
Dependency Requirement Oversight Committee Ids	string	lu.resultColumn.luOptionalOversightCommitteeIds
Dependency Requirement Oversight Committee Names	string	lu.resultColumn.luOptionalOversightCommitteeNames
Different Admin Orgs	boolean	lu.resultColumn.luOptionalDependencyRequirementDifferentAdminOrg

### JPQL Implementation

```
null
```

### Basic and Advanced Search (lu.search.current)

Search Type Key	lu.search.current
Name	Basic and Advanced Search
Description	Query with multiple optional elements to satisfy most advanced pickers

### 14 Possible Search Criteria

Type Key	lu.criteria.luAdvancedCriteria
Name	LuAdvancedCriteria
Description	Advanced criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
CLU Identifier	true	string	true	luQueryParam.luOptionalId	clu.id = :lu_queryParam_luOptionalId
Short Name	true	string	false	luQueryParam.luOptionalShortName	clu.optionalIdentifier.shortName

Long Name	true	string	false	luQueryParam.luOptionalLongName	clu.officeId.alide ntifier.longName
Learning Unit Type	true	string	false	luQueryParam.luOptionalType	clu.uType.id IN (:lu_queryParam_luOptionalType)
Code	true	string	false	luQueryParam.luOptionalCode	LOWER(clu.officeId.alide ntifier.code) LIKE '%'    LOWER(:lu_queryParam_luOptionalCode)    '%'
Division	true	string	false	luQueryParam.luOptionalDivision	clu.officeId.alide ntifier.division

Level	true	string	false	luQueryParam.luOptionalLevel	clu.offici.alIden.tifi.er.level
Clu Description	true	string	false	luQueryParam.luOptionalDescr	clu.descri plain
Study Subject Area	true	string	false	luQueryParam.luOptionalStudySubjectArea	clu.offici.alIden.tifi.er.di vision
Learning Unit State	true	string	false	luQueryParam.luOptionalState	clu.state IN (:lu_queryParam_luOptionalState)
Suffix Code	true	string	false	luQueryParam.luOptionalCrsNoRange	! !NUM BER_R ANGE clu.offici.alIden.tifi.er.suffixCode

Effective Date	true	date	false	lu.queryParam.luOptionalEffectiveDate1	clue ffect iveDa te >= :lu_q ueryP aram_ luOpt ional Effec tiveD ate1
Effective Date	true	date	false	lu.queryParam.luOptionalEffectiveDate2	clue ffect iveDa te <= :lu_q ueryP aram_ luOpt ional Effec tiveD ate2

Version Independent Id	true	string	false	lu.queryParam.luOptionalVersionIndId	<pre> clu.v ersio n.ver sionI ndId = :lu_q ueryP aram_ luOpt ional Versi onInd Id  AND (clu. versi on.cu rrent Versi onSta rt &lt;= CURRE NT_TI MESTA MP AND (clu. versi on.cu rrent Versi onEnd &gt; CURRE NT_TI MESTA MP OR clu.v ersio n.cur rentV ersio nEnd IS NULL) ) </pre>
------------------------	------	--------	-------	--------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 12 Result Columns Returned

Type Key	lu.result.generic
Name	Lu Full View
Description	Full view of clus

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluId
Name	string	lu.resultColumn.luOptionalShortName
Name	string	lu.resultColumn.luOptionalLongName
Code	string	lu.resultColumn.luOptionalCode
Level	string	lu.resultColumn.luOptionalLevel
Description	string	lu.resultColumn.luOptionalDescr
Study Subject Area	string	lu.resultColumn.luOptionalStudySubjectArea
State	string	lu.resultColumn.luOptionalState
VersionIndId	string	lu.resultColumn.luOptionalVersionIndId
VersionedFromId	string	lu.resultColumn.luOptionalVersionedFromId
VersionStartDate	date	lu.resultColumn.luOptionalVersionStartDate
VersionEndDate	date	lu.resultColumn.luOptionalVersionEndDate

### JPQL Implementation

```

SELECT clu.id, officialIdentifier.shortName, officialIdentifier.longName,
       officialIdentifier.code, officialIdentifier.level,
       cludesc.plain, clu.studySubjectArea, clu.state,
       clu.version.versionIndId, clu.version.versionedFromId,
       clu.version.currentVersionStart, clu.version.currentVersionEnd
  FROM Clu clu
  LEFT JOIN clu.officialIdentifier officialIdentifier
  LEFT JOIN clu.descr cludesc
 WHERE (clu.version.currentVersionStart <= CURRENT_TIMESTAMP AND
       (clu.version.currentVersionEnd > CURRENT_TIMESTAMP OR clu.version.currentVersionEnd IS
        NULL) )

```

### Basic and Advanced Search (lu.search.noncurrent)

Search Type Key	lu.search.noncurrent
Name	Basic and Advanced Search
Description	Query with multiple optional elements to satisfy most advanced pickers

### 14 Possible Search Criteria

Type Key	lu.criteria.luAdvancedCriteria
Name	LuAdvancedCriteria
Description	Advanced criteria

Name	Optional	DataType	Read Only	Type Key	Implementation

CLU Identifier	true	string	true	lu.queryParam.luOptionalId	clu.i d = :lu_q ueryP aram_ luOpt ional Id
Short Name	true	string	false	lu.queryParam.luOptionalShortName	clu.o ffici alIde ntifi er.sh ortNa me
Long Name	true	string	false	lu.queryParam.luOptionalLongName	clu.o ffici alIde ntifi er.lo ngNam e
Learning Unit Type	true	string	false	lu.queryParam.luOptionalType	clu.l uType .id IN (:lu_ query Param _luOp tiona lType )

Code	true	string	false	luQueryParam.luOptionalCode	<pre> LOWER (clu. offic ialId entif ier.c ode) LIKE '%'    LOWER (:lu_ query Param _luOp tiona lCode )    '%' </pre>
Division	true	string	false	luQueryParam.luOptionalDivision	<pre> clu.o ffici alIde ntifi er.di visio n </pre>
Level	true	string	false	luQueryParam.luOptionalLevel	<pre> clu.o ffici alIde ntifi er.le vel </pre>
Clu Description	true	string	false	luQueryParam.luOptionalDescr	<pre> clu.d escr. plain </pre>

Study Subject Area	true	string	false	luQueryParam.luOptionalStudySubjectArea	clu.offici alIde ntifi er.di visio n
Learning Unit State	true	string	false	luQueryParam.luOptionalState	clu.state IN (:lu_queryParam_luOptionalStat e)
Suffix Code	true	string	false	luQueryParam.luOptionalCrsNoRange	! !NUM BER_R ANGE clu.offici alIde ntifi er.su ffixCode
Effective Date	true	date	false	luQueryParam.luOptionalEffectiveDate1	clu.e ffec tiveDa te >= :lu_q ueryP aram_ luOpt ional EffectiveD atel

Effective Date	true	date	false	lu.queryParam.luOptionalEffectiveDate2	clue.e ffect iveDa te <= :lu_q ueryP aram_ luOpt ional Effec tiveD ate2
----------------	------	------	-------	----------------------------------------	----------------------------------------------------------------------------------------------------------

Version Independent Id	true	string	false	lu.queryParam.luOptionalVersionIndId	<pre> clu.v ersio n.ver sionI ndId = :lu_q ueryP aram_ luOpt ional Versi onInd Id  AND (clu. versi on.cu rrent Versi onSta rt &lt;= CURRE NT_TI MESTA MP AND (clu. versi on.cu rrent Versi onEnd &gt; CURRE NT_TI MESTA MP OR clu.v ersio n.cur rentV ersio nEnd IS NULL) ) </pre>
------------------------	------	--------	-------	--------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 12 Result Columns Returned

Type Key	lu.result.generic
Name	Lu Full View
Description	Full view of clus

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluId
Name	string	lu.resultColumn.luOptionalShortName
Name	string	lu.resultColumn.luOptionalLongName
Code	string	lu.resultColumn.luOptionalCode
Level	string	lu.resultColumn.luOptionalLevel
Description	string	lu.resultColumn.luOptionalDescr
Study Subject Area	string	lu.resultColumn.luOptionalStudySubjectArea
State	string	lu.resultColumn.luOptionalState
VersionIndId	string	lu.resultColumn.luOptionalVersionIndId
VersionedFromId	string	lu.resultColumn.luOptionalVersionedFromId
VersionStartDate	date	lu.resultColumn.luOptionalVersionStartDate
VersionEndDate	date	lu.resultColumn.luOptionalVersionEndDate

### JPQL Implementation

```

SELECT clu.id, officialIdentifier.shortName, officialIdentifier.longName,
       officialIdentifier.code, officialIdentifier.level,
       cludesc.plain, clu.studySubjectArea, clu.state,
       clu.version.versionIndId, clu.version.versionedFromId,
       clu.version.currentVersionStart, clu.version.currentVersionEnd
  FROM Clu clu
 LEFT JOIN clu.descr cludesc
 LEFT JOIN clu.officialIdentifier officialIdentifier
 WHERE clu.version.sequenceNumber=
       (SELECT MAX(mostrecentclu.version.sequenceNumber)
      FROM Clu mostrecentclu
     LEFT JOIN mostrecentclu.officialIdentifier officialIdentifier
      WHERE clu.version.versionIndId=mostrecentclu.version.versionIndId
      AND mostrecentclu.version.versionIndId IS NOT NULL
      AND mostrecentclu.version.versionIndId NOT IN (
          SELECT activeclu.version.versionIndId
         FROM Clu activeclu
        WHERE activeclu.version.versionIndId=mostrecentclu.version.versionIndId
          AND activeclu.version.versionIndId=mostrecentclu.version.versionIndId
          AND (activeclu.version.currentVersionStart <= CURRENT_DATE AND
               (activeclu.version.currentVersionEnd > CURRENT_DATE OR
                activeclu.version.currentVersionEnd IS NULL))
      )
     GROUP BY mostrecentclu.version.versionIndId
   )

```

### Basic and Advanced Search (lu.search.generic)

Search Type Key	lu.search.generic
Name	Basic and Advanced Search

<b>Description</b>	Query with multiple optional elements to satisfy most advanced pickers
--------------------	------------------------------------------------------------------------

#### 14 Possible Search Criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
CLU Identifier	true	string	true	lu.queryParam.luOptionalId	clu.i d = :lu_q ueryP aram_ luOpt ional Id
Short Name	true	string	false	lu.queryParam.luOptionalShortName	clu.o ffici alIde ntifi er.sh ortNa me
Long Name	true	string	false	lu.queryParam.luOptionalLongName	clu.o ffici alIde ntifi er.lo ngNam e
Learning Unit Type	true	string	false	lu.queryParam.luOptionalType	clu.l uType .id IN (:lu_ query Param _luOp tiona lType )

Code	true	string	false	luQueryParam.luOptionalCode	<pre> LOWER (clu. offic ialId entif ier.c ode) LIKE '%'    LOWER (:lu_ query Param _luOp tiona lCode )    '%' </pre>
Division	true	string	false	luQueryParam.luOptionalDivision	<pre> clu.o ffici alIde ntifi er.di visio n </pre>
Level	true	string	false	luQueryParam.luOptionalLevel	<pre> clu.o ffici alIde ntifi er.le vel </pre>
Clu Description	true	string	false	luQueryParam.luOptionalDescr	<pre> clu.d escr. plain </pre>

Study Subject Area	true	string	false	luQueryParam.luOptionalStudySubjectArea	clu.offici alIde ntifi er.di visio n
Learning Unit State	true	string	false	luQueryParam.luOptionalState	clu.state IN (:lu_queryParam_luOptionalStat e)
Suffix Code	true	string	false	luQueryParam.luOptionalCrsNoRange	! !NUM BER_R ANGE clu.offici alIde ntifi er.su ffixCode
Effective Date	true	date	false	luQueryParam.luOptionalEffectiveDate1	clu.e ffec tiveDa te >= :lu_q ueryP aram_ luOpt ional EffectiveD atel

Effective Date	true	date	false	lu.queryParam.luOptionalEffectiveDate2	clue.e ffect iveDa te <= :lu_q ueryP aram_ luOpt ional Effec tiveD ate2
----------------	------	------	-------	----------------------------------------	----------------------------------------------------------------------------------------------------------

Version Independent Id	true	string	false	lu.queryParam.luOptionalVersionIndId	<pre> clu.v ersio n.ver sionI ndId = :lu_q ueryP aram_ luOpt ional Versi onInd Id  AND (clu. versi on.cu rrent Versi onSta rt &lt;= CURRE NT_TI MESTA MP AND (clu. versi on.cu rrent Versi onEnd &gt; CURRE NT_TI MESTA MP OR clu.v ersio n.cur rentV ersio nEnd IS NULL) ) </pre>
------------------------	------	--------	-------	--------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 12 Result Columns Returned

Type Key	lu.result.generic
Name	Lu Full View
Description	Full view of clus

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluId
Name	string	lu.resultColumn.luOptionalShortName
Name	string	lu.resultColumn.luOptionalLongName
Code	string	lu.resultColumn.luOptionalCode
Level	string	lu.resultColumn.luOptionalLevel
Description	string	lu.resultColumn.luOptionalDescr
Study Subject Area	string	lu.resultColumn.luOptionalStudySubjectArea
State	string	lu.resultColumn.luOptionalState
VersionIndId	string	lu.resultColumn.luOptionalVersionIndId
VersionedFromId	string	lu.resultColumn.luOptionalVersionedFromId
VersionStartDate	date	lu.resultColumn.luOptionalVersionStartDate
VersionEndDate	date	lu.resultColumn.luOptionalVersionEndDate

### JPQL Implementation

```

SELECT clu.id, officialIdentifier.shortName, officialIdentifier.longName,
       officialIdentifier.code, officialIdentifier.level,
       cludesc.plain, clu.studySubjectArea, clu.state,
       clu.version.versionIndId, clu.version.versionedFromId,
       clu.version.currentVersionStart, clu.version.currentVersionEnd
  FROM Clu clu
  LEFT JOIN clu.officialIdentifier officialIdentifier
  LEFT JOIN clu.descr cludesc

```

### Lum Search for Clus (lu.search.clus)

Search Type Key	lu.search.clus
Name	Lum Search for Clus
Description	Returns all available Clus

### 0 Possible Search Criteria

Type Key	lu.criteria.all
Name	LuClu
Description	Placeholder Criteria

Name	Optional	DataType	Read Only	Type Key	Implementation

### 3 Result Columns Returned

Type Key	lu.result.luClus
Name	Lu Clu
Description	List Lu Clus

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluId
Clu Long Name	string	lu.resultColumn.cluOfficialIdentifier.longName
VersionIndId	string	lu.resultColumn.luOptionalVersionIndId

#### JPQL Implementation

```
SELECT clu.id, clu.officialIdentifier.longName, clu.version.versionIndId FROM Clu clu
WHERE LOWER(clu.state) NOT IN ('Draft') ORDER BY
clu.officialIdentifier.longName
```

#### Lum Search for Clus in a Cluset (lu.search.clusInCluset)

Search Type Key	lu.search.clusInCluset
Name	Lum Search for Clus in a Cluset
Description	Returns Clus in a cluset

#### 1 Possible Search Criteria

Type Key	null
Name	null
Description	null

Name	Optional	DataType	Read Only	Type Key	Implementation
Id	true	string	true	cluset.queryParam.optionalId	cluset.id = :cluset_queryParam_optionalId

#### 4 Result Columns Returned

Type Key	lu.result.clusInCluset
Name	Lu Clu
Description	List Lu Clus

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluId
Clu Long Name	string	lu.resultColumn.cluOfficialIdentifier.longName

Clu Code	string	lu.resultColumn.cluOfficialIdentifier.cluCode
VersionIndId	string	lu.resultColumn.luOptionalVersionIndId

### JPQL Implementation

```
SELECT clu.id, clu.officialIdentifier.longName, clu.officialIdentifier.code,
clu.version.versionIndId FROM CluSet cluset
    JOIN cluset.clus clu
```

### Basic and Advanced Search (lu.search.cluCluRelation)

Search Type Key	lu.search.cluCluRelation
Name	Basic and Advanced Search
Description	Query with multiple optional elements to satisfy most advanced pickers

### 1 Possible Search Criteria

Type Key	lu.criteria.cluCluRelation
Name	cluCluRelation
Description	Advanced criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
CLU Identifier	true	string	true	lu.queryParam.luOptionalId	clu.i d = :lu_q ueryP aram_ luOpt ional Id

### 2 Result Columns Returned

Type Key	lu.result.cluCluRelation
Name	Lu Full View
Description	Full view of clus

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluId
Related Clu Id	string	lu.resultColumn.relatedCluId

### JPQL Implementation

```
SELECT rel.clu.id, rel.relatedClu.id FROM CluCluRelation rel
```

## Lum Search for Clus Ids and Codes (lu.search.allLoQuickView)

<b>Search Type Key</b>	lu.search.allLoQuickView
<b>Name</b>	Lum Search for Clus Ids and Codes
<b>Description</b>	Returns all available Clus Ids and Codes

## 8 Possible Search Criteria

<b>Type Key</b>	lu.criteria.allLoQuickView
<b>Name</b>	Clu params for Lo Clus
<b>Description</b>	Clu params for Lo Clus

Name	Optional	DataType	Read Only	Type Key	Implementation
Code	true	string	false	luQueryParam.luOptionalCode	<pre>LOWER (clu. offic ialId entif ier.c ode) LIKE '%'    LOWER (:lu_ query Param _luOp tiona lCode )    '%'</pre>
Level	true	string	false	luQueryParam.luOptionalLevel	<pre>clu.o ffici alIde ntifi er.le vel</pre>

Learning Unit State	true	string	false	lu.queryParam.luOptionalState	clu.state IN (:lu_queryParam_luOptionalState)
Division	true	string	false	lu.queryParam.luOptionalDivision	clu.officeIdentityifier.division
Learning Unit Type	true	string	false	lu.queryParam.luOptionalType	clu.luType.id IN (:lu_queryParam_luOptionalType)
Long Name	true	string	false	lu.queryParam.luOptionalLongName	clu.officeIdentityifier.longName

Admin Org Id	true	string	false	lu.queryParam.luOptionalAdminOrgIds	adminOrg.orgId in (:lu_queryParam_luOptionalAdminOrgIds)
Admin Org Type	true	string	false	lu.queryParam.luOptionalAdminOrgTypes	adminOrg.type in (:lu_queryParam_luOptionalAdminOrgTypes)

## 6 Result Columns Returned

Type Key	lu.result.allLoQuickView
Name	LU Quick View with LO ids
Description	Quick view of the CLU with LO ids

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluId
Clu Type	string	lu.resultColumn.cluType
Clu State	string	lu.resultColumn.cluState
Clu Code	string	lu.resultColumn.cluOfficialIdentifier.cluCode
VersionInId	string	lu.resultColumn.luOptionalVersionInId
Lo Id	string	lu.resultColumn.loId

## JPQL Implementation

```

SELECT DISTINCT clu.id, clu.luType.name, clu.state,
clu.officialIdentifier.code, clu.version.versionIndId, rel.loId
FROM CluLoRelation rel
JOIN rel.clu clu
LEFT JOIN clu.adminOrgs adminOrg

```

### Search for clu id by its code (lu.search.cluByCode)

<b>Search Type Key</b>	lu.search.cluByCode
<b>Name</b>	Search for clu id by its code
<b>Description</b>	Returns clu id

### 0 Possible Search Criteria

<b>Type Key</b>	lu.criteria.code
<b>Name</b>	Code
<b>Description</b>	Clu Code

Name	Optional	DataType	Read Only	Type Key	Implementation

### 1 Result Columns Returned

<b>Type Key</b>	lu.result.cluid
<b>Name</b>	Lu id
<b>Description</b>	Lu id

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluid

### JPQL Implementation

```

SELECT clu.id FROM Clu clu WHERE clu.officialIdentifier.code = :lu_criteria_code ORDER
BY clu.effectiveDate DESC

```

### Code and state match (lu.search.cluByCodeAndState)

<b>Search Type Key</b>	lu.search.cluByCodeAndState
<b>Name</b>	Code and state match
<b>Description</b>	Search by CLU Code and State.

### 2 Possible Search Criteria

<b>Type Key</b>	lu.criteria.cluByCodeAndState

<b>Name</b>	LuByCodeAndStateCriteria				
<b>Description</b>	LuByCodeAndStateCriteria Description				
Name	Optional	DataType	Read Only	Type Key	Implementation
Starts with Clu Code	false	string	true	lu.queryParam.startsWith_cluCode	null
Learning Unit State	false	string	false	lu.queryParam.cluState	null

### 3 Result Columns Returned

<b>Type Key</b>	lu.result.cluQuickView	
<b>Name</b>	LU Quick View	
<b>Description</b>	Quick view of the CLU	

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluId
Clu Code	string	lu.resultColumn.cluOfficialIdentifier.cluCode
VersionIndId	string	lu.resultColumn.luOptionalVersionIndId

### JPQL Implementation

```

SELECT clu.id, clu.officialIdentifier.code, clu.version.versionIndId FROM Clu clu
    WHERE clu.officialIdentifier.code like :lu_queryParam_startsWith_cluCode
    || '%'
        and clu.state = :lu_queryParam_cluState
    ORDER BY clu.officialIdentifier.code ASC

```

### Basic and Advanced Search on Clu Sets (cluset.search.generic)

<b>Search Type Key</b>	cluset.search.generic
<b>Name</b>	Basic and Advanced Search on Clu Sets
<b>Description</b>	Query with multiple optional elements to satisfy most advanced pickers

### 11 Possible Search Criteria

<b>Type Key</b>	cluset.criteria.cluAdvancedCriteria				
<b>Name</b>	Clu Set Advanced Criteria				
<b>Description</b>	Advanced criteria for searching for clu sets				
Name	Optional	DataType	Read Only	Type Key	Implementation

Id	true	string	true	cluset.queryParam.optionalId	cluse t.id = :clus et_queryParam_o ptionalId
Name	true	string	true	cluset.queryParam.optionalName	cluse t.name
Description	true	string	true	cluset.queryParam.optionalDescription	cluse t.description
Long Name	true	string	false	cluset.queryParam.l uOptionalLongName	LOWER (clu offic ialId entif ier.l ongNa me) LIKE '%'    LOWER (:clu set_queryP aram_ luOpt ional LongN ame)    '%' AND clu.ver sionI ndId = cluve rIndI

d.clu  
Versi  
onInd  
Id

AND  
(clu.  
versi  
on.cu  
rrent  
Versi  
onSta  
rt <=  
CURRE  
NT\_TI  
MESTA  
MP  
AND  
(clu.  
versi  
on.cu  
rrent  
Versi  
onEnd  
>  
CURRE  
NT\_TI  
MESTA  
MP OR  
clu.v  
ersio  
n.cur  
rentV  
ersio  
nEnd  
IS

					NULL) )
CLU Identifier	true	string	true	cluset.queryParam.l uOptionalId	

clu.i  
d  
=:clu  
set\_q  
ueryP  
aram\_  
luOpt  
ional  
Id  
AND  
clu.v  
ersio  
n.ver  
sionI  
ndId  
=

cluVe  
rIndI  
d.clu  
Versi  
onInd  
Id  
  
AND  
(clu.  
versi  
on.cu  
rrent  
Versi  
onSta  
rt <=  
CURRE  
NT\_TI  
MESTA  
MP  
AND  
(clu.  
versi  
on.cu  
rrent  
Versi  
onEnd  
>  
CURRE  
NT\_TI  
MESTA  
MP OR  
clu.v  
ersio  
n.cur  
rentV  
ersio  
nEnd  
IS  
NULL)  
)

Code	true	string	false	<pre> cluset.queryParam.l uOptionalCode       LOWER       (clu.       offic       ialId       entif       ier.c       ode)       LIKE       '%'                LOWER       (:clu       set_q       ueryP       aram_       luOpt       ional       Code)                '%'       AND       clu.v       ersio       n.ver       sionI       ndId       =       cluve       rIndI       d.clu       Versi       onInd       Id        AND       (clu.       versi       on.cu       rrent       Versi       onSta       rt &lt;=       CURRE       NT_TI       MESTA       MP       AND       (clu.       versi       on.cu       rrent       Versi       onEnd       &gt;       CURRE       NT_TI       MESTA       MP OR       </pre>
------	------	--------	-------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

clu.v  
ersio  
n.cur  
rentV  
ersio  
nEnd  
IS

					NULL ) )
Clu Set Name	true	string	true	cluset.queryParam.optionalSubCluSetName	subclu set.name
Reusable Clu Set	true	boolean	true	cluset.queryParam.optionalReusable	cluse t.isReusable = :cluset_queryParam_optionalReusable
Referenceable Clu Set	true	boolean	true	cluset.queryParam.optionalReferenceable	cluse t.isReferenceable = :cluset_queryParam_optionalReferenceable
Type	true	string	false	cluset.queryParam.optionalType	cluse t.type = :cluset_queryParam_optionalType

CLU Identifier	true	string	true	lu.queryParam.luOptionalId	clu.id = :lu_queryParam_luOptionalId
----------------	------	--------	------	----------------------------	--------------------------------------

#### 4 Result Columns Returned

Type Key	cluset.result.QuickView
Name	Clu Set Id, Name, Description and Type
Description	Limited view of information in the set

Name	DataType	Type Key
Clu Set Id	string	cluset.resultColumn.cluSetId
Name	string	cluset.resultColumn.name
Description	string	cluset.resultColumn.description
Type	string	cluset.resultColumn.type

#### JPQL Implementation

```
SELECT DISTINCT cluset.id, cluset.name, cluset.descr.plain, cluset.type
  FROM CluSet cluset
  LEFT JOIN cluset.cluVerIndIds cluVerIndID
  LEFT JOIN cluset.cluSets subcluset
  LEFT JOIN cluset.descr clusetDescr
```

#### Basic and Advanced Search on Clu Sets (cluset.search.genericWithClus)

Search Type Key	cluset.search.genericWithClus
Name	Basic and Advanced Search on Clu Sets
Description	Query with multiple optional elements to satisfy most advanced pickers

#### 11 Possible Search Criteria

Type Key	cluset.criteria.cluAdvancedCriteria				
Name	Clu Set Advanced Criteria				
Description	Advanced criteria for searching for clu sets				
Name	Optional	DataType	Read Only	Type Key	Implementation

Id	true	string	true	cluset.queryParam.optionalId	cluset.id = :cluset_queryParam_optionId
Name	true	string	true	cluset.queryParam.optionalName	cluset.name
Description	true	string	true	cluset.queryParam.optionalDescription	cluset.description
Long Name	true	string	false	cluset.queryParam.luOptionalLongName	<pre> LOWER (clu. offic ialId entif ier.l ongNa me) LIKE '%'    LOWER (:clu set_q ueyP aram_ luOpt ional LongN ame)    '%' AND clu.v ersio n.ver sionI ndId = cluve rIndI </pre>

d.clu  
Versi  
onInd  
Id

AND  
(clu.  
versi  
on.cu  
rrent  
Versi  
onSta  
rt <=  
CURRE  
NT\_TI  
MESTA  
MP  
AND  
(clu.  
versi  
on.cu  
rrent  
Versi  
onEnd  
>  
CURRE  
NT\_TI  
MESTA  
MP OR  
clu.v  
ersio  
n.cur  
rentV  
ersio  
nEnd  
IS

					NULL) )
CLU Identifier	true	string	true	cluset.queryParam.l uOptionalId	

clu.i  
d  
=:clu  
set\_q  
ueryP  
aram\_  
luOpt  
ional  
Id  
AND  
clu.v  
ersio  
n.ver  
sionI  
ndId  
=

cluVe  
rIndI  
d.clu  
Versi  
onInd  
Id  
  
AND  
(clu.  
versi  
on.cu  
rrent  
Versi  
onSta  
rt <=  
CURRE  
NT\_TI  
MESTA  
MP  
AND  
(clu.  
versi  
on.cu  
rrent  
Versi  
onEnd  
>  
CURRE  
NT\_TI  
MESTA  
MP OR  
clu.v  
ersio  
n.cur  
rentV  
ersio  
nEnd  
IS  
NULL)  
)

Code	true	string	false	<pre> cluset.queryParam.l uOptionalCode       LOWER       (clu.       offic       ialId       entif       ier.c       ode)       LIKE       '%'                LOWER       (:clu       set_q       ueryP       aram_       luOpt       ional       Code)                '%'       AND       clu.v       ersio       n.ver       sionI       ndId       =       cluve       rIndI       d.clu       Versi       onInd       Id        AND       (clu.       versi       on.cu       rrent       Versi       onSta       rt &lt;=       CURRE       NT_TI       MESTA       MP       AND       (clu.       versi       on.cu       rrent       Versi       onEnd       &gt;       CURRE       NT_TI       MESTA       MP OR </pre>
------	------	--------	-------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

clu.v  
ersio  
n.cur  
rentV  
ersio  
nEnd  
IS

					NULL ) )
Clu Set Name	true	string	true	cluset.queryParam.optionalSubCluSetName	subclu set.name
Reusable Clu Set	true	boolean	true	cluset.queryParam.optionalReusable	cluse t.isReusable = :cluset_queryParam_optionalReusable
Referenceable Clu Set	true	boolean	true	cluset.queryParam.optionalReferenceable	cluse t.isReferenceable = :cluset_queryParam_optionalReferenceable
Type	true	string	false	cluset.queryParam.optionalType	cluse t.type = :cluset_queryParam_optionalType

CLU Identifier	true	string	true	lu.queryParam.luOptionalId	clu.id = :lu_queryParam_luOptionalId
----------------	------	--------	------	----------------------------	--------------------------------------

#### 4 Result Columns Returned

Type Key	cluset.result.QuickView
Name	Clu Set Id, Name, Description and Type
Description	Limited view of information in the set

Name	DataType	Type Key
Clu Set Id	string	cluset.resultColumn.cluSetId
Name	string	cluset.resultColumn.name
Description	string	cluset.resultColumn.description
Type	string	cluset.resultColumn.type

#### JPQL Implementation

```
SELECT DISTINCT cluset.id, cluset.name, cluset.descr.plain, cluset.type
  FROM Clu clu, CluSet cluset
  LEFT JOIN cluset.cluVerIndIds cluVerIndId
  LEFT JOIN cluset.cluSets subcluset
  LEFT JOIN cluset.descr clusetDescr
 WHERE clu.version.versionIndId = cluVerIndId.cluVersionIndId
```

#### Search for clu type optionally by code (lu.search.all.lu.Types)

Search Type Key	lu.search.all.lu.Types
Name	Search for clu type optionally by code
Description	Returns clu name

#### 2 Possible Search Criteria

Type Key	lu.criteria.luType
Name	Lu Type
Description	Lu Type

Name	Optional	DataType	Read Only	Type Key	Implementation
------	----------	----------	-----------	----------	----------------

Lu Type Key	true	string	true	lu.queryParam.luOptionalLuType	lut.id = :lu_queryParam_luOptionalLuType
Lu Type Key	true	string	true	lu.queryParam.luOptionalLuTypeStartsWith	lut.id

## 5 Result Columns Returned

Type Key	lu.result.luTypes
Name	Lu Type
Description	Lu Type

Name	DataType	Type Key
Key of the type	string	lu.resultColumn.luTypeKey
Lu Type Name	string	lu.resultColumn.luTypeName
Lu Type Desc	string	lu.resultColumn.luTypeDesc
Lu Type Effective Date	date	lu.resultColumn.luTypeEffDt
Lu Type Expiration Date	date	lu.resultColumn.luTypeExpDt

## JPQL Implementation

```
SELECT lut.id, lut.name, lut.descr, lut.effectiveDate, lut.expirationDate FROM LuType
lut
```

## Find all versions of a clu (lu.search.clu.versions)

Search Type Key	lu.search.clu.versions
Name	Find all versions of a clu
Description	Query to find all versions of a Clu

## 1 Possible Search Criteria

Type Key	lu.criteria.cluVersionCriteria
Name	Version Independent Id
Description	Version Independent Id

Name	Optional	DataType	Read Only	Type Key	Implementation
Version Independent Id	false	string	true	lu.queryParam.cluVersionIndId	null

## 16 Result Columns Returned

Type Key	lu.result.clu.versions
Name	Clu Versions
Description	Clu Versions

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluId
Name	string	lu.resultColumn.luOptionalShortName
Name	string	lu.resultColumn.luOptionalLongName
Code	string	lu.resultColumn.luOptionalCode
Level	string	lu.resultColumn.luOptionalLevel
Description	string	lu.resultColumn.luOptionalDescr
Study Subject Area	string	lu.resultColumn.luOptionalStudySubjectArea
State	string	lu.resultColumn.luOptionalState
Expected First Atp	string	lu.resultColumn.luOptionalExpFirstAtp
Last Atp	string	lu.resultColumn.luOptionalLastAtp
VersionedFromId	string	lu.resultColumn.luOptionalVersionedFromId
VersionStartDate	date	lu.resultColumn.luOptionalVersionStartDate
VersionEndDate	date	lu.resultColumn.luOptionalVersionEndDate
VersionSeqNum	int	lu.resultColumn.luOptionalVersionSeqNum
Expected First Atp	string	lu.resultColumn.luOptionalExpFirstAtpDisplay
Last Atp	string	lu.resultColumn.luOptionalLastAtpDisplay

## JPQL Implementation

```

SELECT clu.id, officialIdentifier.shortName, officialIdentifier.longName,
       officialIdentifier.code, officialIdentifier.level,
       cludesc.plain, clu.studySubjectArea, clu.state,
       clu.expectedFirstAtp, clu.lastAtp,
       clu.version.versionedFromId, clu.version.currentVersionStart,
       clu.version.currentVersionEnd, clu.version.sequenceNumber, clu.expectedFirstAtp,
       clu.lastAtp
  FROM Clu clu
  LEFT JOIN clu.officialIdentifier officialIdentifier
  LEFT JOIN clu.descr cludesc
 WHERE clu.version.versionIndId = :lu_queryParam_cluVersionIndId

```

## Basic Search for Publication Types (lu.search.publication.types)

<b>Search Type Key</b>	lu.search.publication.types
<b>Name</b>	Basic Search for Publication Types
<b>Description</b>	Return list of Publication Types

### 2 Possible Search Criteria

<b>Type Key</b>	lu.criteria.publicationType
<b>Name</b>	Publication Type Criteria
<b>Description</b>	Publication Type criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
Publication Type Key	true	string	true	lu.queryParam.publicationType.id	pub.id = :lu_queryParam_publicationType_id
Publication Type Ids Key	true	string	true	lu.queryParam.publicationType.idRestrictionList	pub.id IN (:lu_queryParam_publicationType_idRestrictionList)

### 2 Result Columns Returned

<b>Type Key</b>	lu.result.publicationType
<b>Name</b>	publication Type View
<b>Description</b>	View of publication Types

Name	DataType	Type Key
Publication Type Key	string	lu.resultColumn.publicationType.id
Publication Type Name	string	lu.resultColumn.publicationType.name

### JPQL Implementation

```

SELECT pub.id, pub.name, pub.descr, pub.effectiveDate, pub.expirationDate FROM
CluPublicationType pub

```

## LU Search for Majors and Variations (lu.search.union.majors)

<b>Search Type Key</b>	lu.search.union.majors
<b>Name</b>	LU Search for Majors and Variations
<b>Description</b>	Returns all matching Majors and Variations

### 6 Possible Search Criteria

<b>Type Key</b>	lu.criteria.union.majors
<b>Name</b>	LuAdvancedCriteria
<b>Description</b>	Advanced criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
CLU Identifier	true	string	true	luQueryParam.luOptionalId	clu.i d = :lu_q ueryP aram_ luOpt ional Id
Long Name	true	string	false	luQueryParam.luOptionalLongName	clu.o ffici alIde ntifi er.lo ngNam e

Code	true	string	false	lu.queryParam.luOptionalCode	<pre> LOWER (clu. offic ialId entif ier.c ode) LIKE '%'    LOWER (:lu_ query Param _luOp tiona lCode )    '%' </pre>
Learning Unit State	true	string	false	lu.queryParam.luOptionalSingleState	<pre> lower (clu. state ) like '%'    LOWER (:lu_ query Param _luOp tiona lSing leSta te)     '%' </pre>

Learning Unit State	true	string	false	lu.queryParam.luOptionalCurrentVersion	(clu. versi on.cu rrent Versi onSta rt <= CURRE NT_DA TE AND (clu. versi on.cu rrent Versi onEnd > CURRE NT_DA TE OR clu.v ersio n.cur rentV ersio nEnd IS NULL)
					AND '1'=: lu_qu eryPa ram_l uOpti onalC urren tVers ion)

CLU Identifier Identifier	true	string	true	lu.queryParam.luOptionalCredentials	credit ident. code = :lu_queryP aram_ luOpt ional Crede ntial s
------------------------------	------	--------	------	-------------------------------------	--------------------------------------------------------------------------------------------

#### 8 Result Columns Returned

Type Key	lu.result.union.majors
Name	Lu Full View
Description	Full view of clus

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluld
Name	string	lu.resultColumn.variationId
Code	string	lu.resultColumn.luOptionalCode
State	string	lu.resultColumn.luOptionalState
Name	string	lu.resultColumn.luOptionalMajorName
Name	string	lu.resultColumn.luOptionalVariationName
Lu Type	string	lu.resultColumn.luOptionalType
Name	string	lu.resultColumn.luOptionalShortName

#### JPQL Implementation

```
null
```

#### Search for Majors (lu.search.luProgramsByName)

Search Type Key	lu.search.luProgramsByName
Name	Search for Majors
Description	Search returns majors matching longname

#### 6 Possible Search Criteria

Type Key	lu.criteria.luProgramsByName
Name	LuAdvancedCriteria
Description	Advanced criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
CLU Identifier	true	string	true	luQueryParam.luOptionalId	<pre> clu.i d = :lu_q ueryP aram_ luOpt ional Id </pre>
Code	true	string	false	luQueryParam.luOptionalCode	<pre> LOWER (clu. offic ialId entif ier.c ode) LIKE '%'    LOWER (:lu_ query Param _luOp tiona lCode )    '%' </pre>
Long Name	true	string	false	luQueryParam.luOptionalLongName	<pre> clu.o ffici alIde ntifi er.lo ngNam e </pre>

Learning Unit State	true	string	false	lu.queryParam.luOptionalSingleState	<pre> lower (clu. state ) like '%'    LOWER (:lu_ query Param _luOp tiona lsing leSta te)    '%' </pre>
---------------------	------	--------	-------	-------------------------------------	---------------------------------------------------------------------------------------------------------

Learning Unit State	true	string	false	lu.queryParam.luOptionalCurrentVersion	(clu. versi on.cu rrent Versi onSta rt <= CURRE NT_DA TE AND (clu. versi on.cu rrent Versi onEnd > CURRE NT_DA TE OR clu.v ersio n.cur rentV ersio nEnd IS NULL)
					AND '1'=: lu_qu eryPa ram_l uOpti onalC urren tVers ion)

CLU Identifier Identifier	true	string	true	lu.queryParam.luOptionalCredentials	credient. ident. code = :lu_queryParam_luOptionalCredentials
------------------------------	------	--------	------	-------------------------------------	-----------------------------------------------------------------------

#### 8 Result Columns Returned

Type Key	lu.result.luProgramsByName
Name	Lu Full View
Description	Full view of clus

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluId
Name	string	lu.resultColumn.variationId
Code	string	lu.resultColumn.luOptionalCode
State	string	lu.resultColumn.luOptionalState
Lu Type	string	lu.resultColumn.luOptionalType
Name	string	lu.resultColumn.luOptionalMajorName
Name	string	lu.resultColumn.luOptionalLongName
Name	string	lu.resultColumn.luOptionalShortName

#### JPQL Implementation

```

SELECT clu.id, ' ', ident1.code, clu.state, clu.luType.id,
ident1.longName, 'N/A', credident.shortName
FROM Clu clu, CluIdentifier ident1, Clu credential,
CluIdentifier credident, CluCluRelation relation
WHERE clu.officialIdentifier.id = ident1.id
AND clu.luType.id = 'kuali.lu.type.MajorDiscipline'
AND clu.id = relation.relatedClu.id
AND relation.clu.id = credential.id
AND credential.officialIdentifier.id = credident.id
AND relation.luLuRelationType.id =
'kuali.lu.lu.relation.type.hasMajorProgram'

```

#### Search for Majors (lu.search.luVariationsByName)

<b>Search Type Key</b>	lu.search.luVariationsByName
<b>Name</b>	Search for Majors
<b>Description</b>	Search returns majors matching longname

## 6 Possible Search Criteria

<b>Type Key</b>	lu.criteria.luVariationsByName
<b>Name</b>	LuAdvancedCriteria
<b>Description</b>	Advanced criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
CLU Identifier	true	string	true	lu.queryParam.luOptionalId	<pre> clu.i d = :lu_q ueryP aram_ luOpt ional Id </pre>
Long Name	true	string	false	lu.queryParam.luOptionalLongName	<pre> clu.o ffici alIde ntifi er.lo ngNam e </pre>
Code	true	string	false	lu.queryParam.luOptionalCode	<pre> LOWER (clu. offic ialId entif ier.c ode) LIKE '%'      LOWER (:lu_ query Param _luOp tiona lCode )     '%' </pre>

Learning Unit State	true	string	false	lu.queryParam.luOptionalSingleState	<pre> lower (clu. state ) like '%'    LOWER (:lu_ query Param _luOp tiona lsing leSta te)    '%' </pre>
---------------------	------	--------	-------	-------------------------------------	---------------------------------------------------------------------------------------------------------

Learning Unit State	true	string	false	lu.queryParam.luOptionalCurrentVersion	(clu. versi on.cu rrent Versi onSta rt <= CURRE NT_DA TE AND (clu. versi on.cu rrent Versi onEnd > CURRE NT_DA TE OR clu.v ersio n.cur rentV ersio nEnd IS NULL)
					AND '1'=: lu_qu eryPa ram_l uOpti onalC urren tVers ion)

CLU Identifier Identifier	true	string	true	lu.queryParam.luOptionalCredentials	credient. code = :lu_queryParam_luOptional_Credentials
------------------------------	------	--------	------	-------------------------------------	--------------------------------------------------------------

#### 8 Result Columns Returned

Type Key	lu.result.luVariationsByName
Name	Lu Full View
Description	Full view of clus

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluld
Name	string	lu.resultColumn.variationId
Code	string	lu.resultColumn.luOptionalCode
State	string	lu.resultColumn.luOptionalState
Lu Type	string	lu.resultColumn.luOptionalType
Name	string	lu.resultColumn.luOptionalMajorName
Name	string	lu.resultColumn.luOptionalLongName
Name	string	lu.resultColumn.luOptionalShortName

#### JPQL Implementation

```

        SELECT clu2.id, clu.id, ident1.code, clu.state,
clu.luType.id, ident2.longName, ident1.longName, credident.shortName
        FROM Clu clu, CluIdentifier ident1, Clu clu2, CluIdentifier
ident2, CluCluRelation rel, Clu credential ,CluIdentifier credident , CluCluRelation
rel2
        WHERE clu.id = rel.relatedClu.id      and rel.clu.id = clu2.id
        and clu.officialIdentifier.id = ident1.id
        and clu2.officialIdentifier.id = ident2.id
        AND rel.luLuRelationType.id =
'kuali.lu.lu.relation.type.hasVariationProgram'
        AND clu.luType.id = 'kuali.lu.type.Variation'
        AND clu2.id = rel2.relatedClu.id
        AND rel2.clu.id = credential.id
        AND credential.officialIdentifier.id = credident.id
        AND rel2.luLuRelationType.id =
'kuali.lu.lu.relation.type.hasMajorProgram'

```

## Find clu's based on the admin org (lu.search.by.admin.org)

<b>Search Type Key</b>	lu.search.by.admin.org
<b>Name</b>	Find clu's based on the admin org
<b>Description</b>	Intended for searches that need to find clu's associated with organizations

## 15 Possible Search Criteria

<b>Type Key</b>	lu.criteria.admin.org
<b>Name</b>	Search by admin org and type
<b>Description</b>	Search by the admin or and the type of relationship the org has to the clu

<b>Name</b>	<b>Optional</b>	<b>DataType</b>	<b>Read Only</b>	<b>Type Key</b>	<b>Implementation</b>
CLU Identifier	true	string	true	lu.queryParam.luOptionalId	clu.i d = :lu_q ueryP aram_ luOpt ional Id

Short Name	true	string	false	luQueryParam.luOptionalShortName	clu.offici alId entifi er.sh ortNa me
Long Name	true	string	false	luQueryParam.luOptionalLongName	clu.offici alId entifi er.lo ngNam e
Learning Unit Type	true	string	false	luQueryParam.luOptionalType	clu.l uType .id IN (:lu_queryParam_luOpt ionalType )
Code	true	string	false	luQueryParam.luOptionalCode	LOWER (clu offic ialId entifi er.c ode) LIKE '%'    LOWER (:lu_queryParam_luOpt ionalCode )    '%'

Division	true	string	false	luQueryParam.luOptionalDivision	clu.offici alIde ntifi er.di visio n
Level	true	string	false	luQueryParam.luOptionalLevel	clu.offici alIde ntifi er.level
Clu Description	true	string	false	luQueryParam.luOptionalDescr	clu.d escr. plain
Study Subject Area	true	string	false	luQueryParam.luOptionalStudySubjectArea	clu.offici alIde ntifi er.di visio n
Admin Org Id	true	string	false	luQueryParam.luOptionalAdminOrgIds	admin.Org.o rgId in (:lu_queryParam._luOptionalAdmi nOrgIds)

Admin Org Type	true	string	false	luQueryParam.luOptionalAdminOrgTypes	admin Org.t ype in (:lu_ query Param _luOp tiona lAdmi nOrgT ypes)
Learning Unit State	true	string	false	luQueryParam.luOptionalState	clu.s tate IN (:lu_ query Param _luOp tiona lStat e)
Suffix Code	true	string	false	luQueryParam.luOptionalCrsNoRange	! !NUM BER_R ANGE clu.o ffici alIde ntifi er.su ffixC ode
Effective Date	true	date	false	luQueryParam.luOptionalEffectiveDate1	clu.e ffect iveDa te >= :lu_q ueryP aram_ luOpt ional Effec tiveD ate1

Effective Date	true	date	false	lu.queryParam.luOptionalEffectiveDate2	clu.e ffect iveDa te <= :lu_q ueryP aram_ luOpt ional Effec tiveD ate2
----------------	------	------	-------	----------------------------------------	---------------------------------------------------------------------------------------------------------

#### 14 Result Columns Returned

Type Key	lu.result.admin.org
Name	Full View of Clu plus admin org info
Description	All information in the full view plus admin org id and type

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluId
Name	string	lu.resultColumn.luOptionalShortName
Name	string	lu.resultColumn.luOptionalLongName
Code	string	lu.resultColumn.luOptionalCode
Level	string	lu.resultColumn.luOptionalLevel
Description	string	lu.resultColumn.luOptionalDescr
Study Subject Area	string	lu.resultColumn.luOptionalStudySubjectArea
State	string	lu.resultColumn.luOptionalState
VersionIndId	string	lu.resultColumn.luOptionalVersionIndId
VersionedFromId	string	lu.resultColumn.luOptionalVersionedFromId
VersionStartDate	date	lu.resultColumn.luOptionalVersionStartDate
VersionEndDate	date	lu.resultColumn.luOptionalVersionEndDate
AdminOrgId	String	lu.resultColumn.luOptionalAdminOrgId
Administrative Org Type	String	lu.resultColumn.luOptionalAdminOrgType

#### JPQL Implementation

```

SELECT clu.id, officialIdentifier.shortName, officialIdentifier.longName,
       officialIdentifier.code, officialIdentifier.level,
       cludesc.plain, clu.studySubjectArea, clu.state,
       clu.version.versionIndId, clu.version.versionedFromId,
       clu.version.currentVersionStart, clu.version.currentVersionEnd,
       adminOrg.orgId, adminOrg.type
  FROM CluAdminOrg adminOrg
 LEFT JOIN adminOrg.clu clu
 LEFT JOIN clu.officialIdentifier officialIdentifier
 LEFT JOIN clu.descr cludesc

```

## Search for Majors (lu.search.luByRelation)

<b>Search Type Key</b>	lu.search.luByRelation
<b>Name</b>	Search for Majors
<b>Description</b>	Search returns majors matching longname

## 2 Possible Search Criteria

<b>Type Key</b>	lu.criteria.luByRelation
<b>Name</b>	LuAdvancedCriteria
<b>Description</b>	Advanced criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
Related Clu Id	true	String	false	lu.queryParam.luOptionalRelatedCluId	rel.relate dClu.id = :lu_queryParam_luOptionalRelatedCluId

Relation Type	true	String	false	lu.queryParam.luOptionalRelationType	rel.1 uLuRe latio nType .id = :lu_q ueryP aram_ luOpt ional Relat ionTy pe
---------------	------	--------	-------	--------------------------------------	----------------------------------------------------------------------------------------------------------------

### 13 Result Columns Returned

Type Key	lu.result.luByRelation
Name	Lu Full View
Description	Full view of clus

Name	DataType	Type Key
Clu Id	string	lu.resultColumn.cluld
Name	string	lu.resultColumn.luOptionalShortName
Name	string	lu.resultColumn.luOptionalLongName
Code	string	lu.resultColumn.luOptionalCode
Level	string	lu.resultColumn.luOptionalLevel
Description	string	lu.resultColumn.luOptionalDescr
Study Subject Area	string	lu.resultColumn.luOptionalStudySubjectArea
State	string	lu.resultColumn.luOptionalState
VersionIndId	string	lu.resultColumn.luOptionalVersionIndId
VersionedFromId	string	lu.resultColumn.luOptionalVersionedFromId
VersionStartDate	date	lu.resultColumn.luOptionalVersionStartDate
VersionEndDate	date	lu.resultColumn.luOptionalVersionEndDate
Clu CLu Relation Type	String	lu.resultColumn.luOptionalRelationType

### JPQL Implementation

```

SELECT    clu.id,    ident.shortName, ident.longName,
ident.code, ident.level,
          cludesc.plain, clu.studySubjectArea, clu.state,
          clu.version.versionIndId, clu.version.versionedFromId,
clu.version.currentVersionStart, clu.version.currentVersionEnd,
          rel.luLuRelationType.id
FROM Clu clu, CluIdentifier ident, CluCluRelation rel
LEFT JOIN clu.descr cludesc
WHERE rel.clu.id = clu.id
AND clu.officialIdentifier.id = ident.id

```

## (CM 2.0) Formatted View of Comment Searches

 This page was automatically generated on Tue Dec 11 12:23:32 EST 2012  
DO NOT UPDATE MANUALLY!

This page represents a formatted view of `comment-search-config.xml`

- Full view by comment id key (`comment.search.commentTextById`)
  - 1 Possible Search Criteria
  - 2 Result Columns Returned
  - JPQL Implementation
- Full view by tag id key (`tag.search.tagNamespaceById`)
  - 1 Possible Search Criteria
  - 2 Result Columns Returned
  - JPQL Implementation

### Full view by comment id key (`comment.search.commentTextById`)

<b>Search Type Key</b>	<code>comment.search.commentTextById</code>
<b>Name</b>	Full view by comment id key
<b>Description</b>	Returns a full view of comment information for the id supplied

#### 1 Possible Search Criteria

<b>Type Key</b>	<code>comment.criteria.commentById</code>
<b>Name</b>	<code>CommentByIdCriteria</code>
<b>Description</b>	<code>CommentByIdCriteria</code> Description

Name	Optional	DataType	Read Only	Type Key	Implementation
Comment Identifier	false	string	false	<code>comment.queryParam.commentId</code>	null

#### 2 Result Columns Returned

<b>Type Key</b>	<code>comment.result.commentFullView</code>
<b>Name</b>	Comment Text View
<b>Description</b>	Comment Text

Name	DataType	Type Key
Comment Identifier	string	comment.resultColumn.commentId
Comment Plain Text	string	comment.resultColumn.commentText

### JPQL Implementation

```
SELECT comment.id, commentText.plain FROM Comment comment JOIN comment.commentText
commentText WHERE comment.id = :comment_queryParam_commentId
```

### Full view by tag id key (tag.search.tagNamespaceById)

Search Type Key	tag.search.tagNamespaceById
Name	Full view by tag id key
Description	Returns a full view of tag information for the id supplied

### 1 Possible Search Criteria

Type Key	tag.criteria.tagById				
Name	TagByIdCriteria				
Description	TagByIdCriteria Description				
Name	Optional	DataType	Read Only	Type Key	Implementation
id	false	String	false	tag.queryParam.tagId	null

### 2 Result Columns Returned

Type Key	tag.result.tagFullView	
Name	Tag Text View	
Description	Tag Text	

Name	DataType	Type Key
Tag Identifier	string	tag.resultColumn.tagId
Tag Namespace	string	tag.resultColumn.tagNamespace

### JPQL Implementation

```
SELECT tag.id, tag.namespace FROM Tag tag WHERE tag.id = :tag_queryParam_tagId
```

## (CM 2.0) Formatted View of EM Searches

⚠ This page was automatically generated on Tue Dec 11 12:23:32 EST 2012  
DO NOT UPDATE MANUALLY!

This page represents a formatted view of em-search-config.xml

- Get enumerations (enumeration.management.search)
  - 6 Possible Search Criteria
  - 7 Result Columns Returned
  - JPQL Implementation

## Get enumerations (enumeration.management.search)

Search Type Key	enumeration.management.search
Name	Get enumerations
Description	Get enumerated lists of values given context

### 6 Possible Search Criteria

Type Key	enumeration.criteria.context
Name	Enumeration Criteria
Description	Enumeration Criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
Enumeration Type	false	string	false	enumeration.queryParam.enumerationType	null
Context Type	true	string	false	enumeration.queryParam.contextType	enumContext.typeKey
Context Value	true	string	false	enumeration.queryParam.contextValue	LOWER(enumValue.value) LIKE '%  LOWER(:enumeration_queryParam_contextValue)  %'

Context Date	true	dateTime	false	enumeration.queryParam.contextDate	null
Enumeration Code	true	string	false	enumeration.queryParam.enumerationCode	enumValue.value.code IN(:enumeration_queryParam_enumerationType)
Enumeration Code	true	string	false	enumeration.queryParam.enumerationOptionalCode	enumValue.value

## 7 Result Columns Returned

Type Key	enumeration.result.all
Name	All enumeration results
Description	Enumeration Results

Name	DataType	Type Key
Code	string	enumeration.resultColumn.id
Code	string	enumeration.resultColumn.code
Abbreviation of the code	string	enumeration.resultColumn.abbrevValue
Description of the code	string	enumeration.resultColumn.value
Effective date	date	enumeration.resultColumn.effectiveDate
Expiration date	date	enumeration.resultColumn.expirationDate
Key used to sort the results	string	enumeration.resultColumn.sortKey

## JPQL Implementation

```
SELECT DISTINCT enumValue.id, enumValue.code, enumValue.abbrevValue, enumValue.value,
enumValue.effectiveDate, enumValue.expirationDate, enumValue.sortKey FROM
EnumeratedValue enumValue LEFT JOIN enumValue.contextEntityList enumContext WHERE
enumValue.enumeration.id IN(:enumeration_queryParam_enumerationType)
```

## (CM 2.0) Formatted View of LO Searches

**⚠** This page was automatically generated on Tue Dec 11 12:23:38 EST 2012  
DO NOT UPDATE MANUALLY!

This page represents a formatted view of `lo-search-config.xml`

- LO Search for all LOs and related Clus matching supplied word (`lo.search.loByDescCrossSearch`)
  - 1 Possible Search Criteria
  - 4 Result Columns Returned
  - JPQL Implementation
- LO Search for all LOs and related Clus matching category name (`lo.search.loByCategoryCluCrossSearch`)
  - 3 Possible Search Criteria
  - 8 Result Columns Returned
  - JPQL Implementation
- LoSearchByName (`lo.search.loByName`)
  - 1 Possible Search Criteria
  - 2 Result Columns Returned
  - JPQL Implementation
- LO Search for all LOs (`lo.search.loS`)
  - 0 Possible Search Criteria
  - 2 Result Columns Returned
  - JPQL Implementation
- LO Search for all LOs matching supplied word (`lo.search.loByDesc`)
  - 1 Possible Search Criteria
  - 2 Result Columns Returned
  - JPQL Implementation
- LO Search for all LOs and related Clus matching supplied word (`lo.search.loCluByDesc`)
  - 1 Possible Search Criteria
  - 4 Result Columns Returned
  - JPQL Implementation
- LO Search for all LOs by matching category (`lo.search.loByCategory`)
  - 3 Possible Search Criteria
  - 3 Result Columns Returned
  - JPQL Implementation
- LO Search for all LOs by Clu Code (`lo.search.loByCluCode`)
  - 1 Possible Search Criteria
  - 5 Result Columns Returned
  - JPQL Implementation
- LO Search for all LOs by matching category (`lo.search.loByCluCategory`)
  - 1 Possible Search Criteria
  - 5 Result Columns Returned
  - JPQL Implementation
- LO Search for all categories (`lo.search.loCategories`)
  - 3 Possible Search Criteria
  - 6 Result Columns Returned
  - JPQL Implementation
- LO Search for all categories based on name, type and state (`lo.search.loCategoriesByNameRepoTypeState`)
  - 4 Possible Search Criteria
  - 1 Result Columns Returned
  - JPQL Implementation

## LO Search for all LOs and related Clus matching supplied word (`lo.search.loByDescCrossSearch`)

Search Type Key	lo.search.loByDescCrossSearch
Name	LO Search for all LOs and related Clus matching supplied word
Description	Returns all matching LOs ids and related Clu ids and codes

### 1 Possible Search Criteria

Type Key	lo.criteria.loByDesc
Name	LoByWordMatchCriteria
Description	null

Name	Optional	DataType	Read Only	Type Key	Implementation
------	----------	----------	-----------	----------	----------------

Plain Text	true	string	true	lo.queryParam.loDescPlain	lo.descriplain
------------	------	--------	------	---------------------------	----------------

#### 4 Result Columns Returned

Type Key	lo.result.loCluByDesc
Name	Los and Related Clus
Description	Lo desc ids and Clu ids and Codes

Name	DataType	Type Key
Lo Identifier	string	lo.resultColumn.loId
Clu Identifier	string	lo.resultColumn.loCluId
Clu Code	string	lo.resultColumn.loCluCode
Lo Desc	string	lo.resultColumn.loDescPlain

#### JPQL Implementation

null

#### LO Search for all LOs and related Clus matching category name (lo.search.loByCategoryCluCrossSearch)

Search Type Key	lo.search.loByCategoryCluCrossSearch
Name	LO Search for all LOs and related Clus matching category name
Description	Returns all matching LOs ids and related Clu ids and codes

#### 3 Possible Search Criteria

Type Key	lo.criteria.byOptionalCategoryName
Name	LoByCategoryCriteria
Description	null

Name	Optional	DataType	Read Only	Type Key	Implementation
Learning Objective Category Name	true	string	false	lo.queryParam.loOptionalCategoryName	cat.name
Plain Text	true	string	true	lo.queryParam.loDescPlain	lo.descriplain

Group Categories	true	String	false	lo.queryParam.grou pCategories	null
------------------	------	--------	-------	-----------------------------------	------

## 8 Result Columns Returned

Type Key	lo.result.loCluByCategory
Name	Los and Related Clus
Description	Lo desc ids and Clu ids and Codes

Name	DataType	Type Key
Lo Identifier	string	lo.resultColumn.loId
Clu Identifier	string	lo.resultColumn.loCluId
Clu Code	string	lo.resultColumn.loCluCode
Lo Desc	string	lo.resultColumn.loDescPlain
Clu Type	string	lo.resultColumn.loCluType
Clu State	string	lo.resultColumn.loCluState
Category	string	lo.resultColumn.categoryName
Clu Code	string	lu.resultColumn.cluOfficialIdentifier.cluCode

## JPQL Implementation

null

## LoSearchByName (lo.search.loByName)

Search Type Key	lo.search.loByName
Name	LoSearchByName
Description	Retrieve LO by name

## 1 Possible Search Criteria

Type Key	lo.criteria.loName
Name	LoSearchByNameCriteria
Description	Search criteria for searching for learning objectives by name

Name	Optional	DataType	Read Only	Type Key	Implementation
Learning Objective Name	false	string	false	lo.queryParam.loNa me	null

## 2 Result Columns Returned

Type Key	lo.result.los
----------	---------------

<b>Name</b>	Los
<b>Description</b>	List Los

Name	DataType	Type Key
Lo Identifier	string	lo.resultColumn.loId
Lo Name	string	lo.resultColumn.loName

#### JPQL Implementation

```
SELECT lo.id, lo.name FROM Lo lo WHERE lo.name = :lo_queryParam_loName
```

#### LO Search for all LOs (lo.search.los)

<b>Search Type Key</b>	lo.search.los
<b>Name</b>	LO Search for all LOs
<b>Description</b>	Returns all available LOs

#### 0 Possible Search Criteria

<b>Type Key</b>	lo.criteria.all
<b>Name</b>	LoLo
<b>Description</b>	Placeholder Criteria

Name	Optional	DataType	Read Only	Type Key	Implementation

#### 2 Result Columns Returned

<b>Type Key</b>	lo.result.los
<b>Name</b>	Los
<b>Description</b>	List Los

Name	DataType	Type Key
Lo Identifier	string	lo.resultColumn.loId
Lo Name	string	lo.resultColumn.loName

#### JPQL Implementation

```
SELECT lo.id, lo.name FROM Lo lo
```

#### LO Search for all LOs matching supplied word (lo.search.loByDesc)

<b>Search Type Key</b>	lo.search.loByDesc
<b>Name</b>	LO Search for all LOs matching supplied word
<b>Description</b>	Returns all matching LOs

## 1 Possible Search Criteria

Type Key	lo.criteria.loByDesc
Name	LoByWordMatchCriteria
Description	null

Name	Optional	DataType	Read Only	Type Key	Implementation
Plain Text	true	string	true	lo.queryParam.loDescPlain	lo.descriplain

## 2 Result Columns Returned

Type Key	lo.result.loDesc
Name	Los
Description	Lo desc

Name	DataType	Type Key
Desc Identifier	string	lo.resultColumn.loDesId
Lo Desc	string	lo.resultColumn.loDescPlain

## JPQL Implementation

```
SELECT lo.id, lo.descr.plain FROM Lo lo
```

## LO Search for all LOs and related Clus matching supplied word (lo.search.loCluByDesc)

Search Type Key	lo.search.loCluByDesc
Name	LO Search for all LOs and related Clus matching supplied word
Description	Returns all matching LOs ids and related Clu ids and codes

## 1 Possible Search Criteria

Type Key	lo.criteria.loByDesc
Name	LoByWordMatchCriteria
Description	null

Name	Optional	DataType	Read Only	Type Key	Implementation
Plain Text	true	string	true	lo.queryParam.loDescPlain	lo.descriplain

#### 4 Result Columns Returned

Type Key	lo.result.loCluByDesc
Name	Los and Related Clus
Description	Lo desc ids and Clu ids and Codes

Name	DataType	Type Key
Lo Identifier	string	lo.resultColumn.loId
Clu Identifier	string	lo.resultColumn.loCluId
Clu Code	string	lo.resultColumn.loCluCode
Lo Desc	string	lo.resultColumn.loDescPlain

#### JPQL Implementation

```

NATIVE:SELECT
    lo.id ,
    clu.id ,
    officialIdentifier.cd ,
    lodesc.plain
FROM
    KSLU_CLU clu
JOIN
    KSLU_CLU_LO_RELTN jn
    ON
        jn.CLU_ID = clu.ID
JOIN
    KSLO_LO lo
    ON
        jn.LO_ID = lo.ID
JOIN
    KSLU_CLU_IDENT officialIdentifier
    ON
        clu.OFFIC_CLU_ID = officialIdentifier.ID
JOIN
    KSLO_RICH_TEXT_T lodesc
    ON
        lodesc.ID = lo.RT_DESCR_ID

```

#### LO Search for all LOs by matching category (lo.search.loByCategory)

Search Type Key	lo.search.loByCategory
Name	LO Search for all LOs by matching category
Description	Returns all matching LOs with lo id, description and category

#### 3 Possible Search Criteria

Type Key	lo.criteria.byOptionalCategoryName
Name	LoByCategoryCriteria
Description	null

Name	Optional	DataType	Read Only	Type Key	Implementation
Learning Objective Category Name	true	string	false	lo.queryParam.loOptionalCategoryName	cat.name
Plain Text	true	string	true	lo.queryParam.loDescPlain	lo.descplain
Group Categories	true	String	false	lo.queryParam.groupCategories	null

### 3 Result Columns Returned

Type Key	lo.result.loByCategory
Name	Los and Related Clus
Description	Lo desc ids and Clu ids and Codes

Name	DataType	Type Key
Lo Identifier	string	lo.resultColumn.loId
Lo Desc	string	lo.resultColumn.loDescPlain
Category	string	lo.resultColumn.categoryName

### JPQL Implementation

```
SELECT lo.id, lo.descr.plain, cat.name FROM Lo lo LEFT JOIN lo.categories lolocat LEFT JOIN lolocat.loCategory cat
```

### LO Search for all LOs by Clu Code (lo.search.loByCluCode)

Search Type Key	lo.search.loByCluCode
Name	LO Search for all LOs by Clu Code
Description	Returns all matching LOs ids

### 1 Possible Search Criteria

Type Key	lo.criteria.byCluCode
Name	LoByCategoryCriteria
Description	null

Name	Optional	DataType	Read Only	Type Key	Implementation

Course Code	true	string	true	lo.queryParam.cluOfficialIdentifier.cluCode	officealId
-------------	------	--------	------	---------------------------------------------	------------

#### 5 Result Columns Returned

Type Key	lo.result.loClu
Name	Los and Related Clus
Description	Lo descs ids and Clu ids and Codes

Name	DataType	Type Key
Lo Identifier	string	lo.resultColumn.loId
Clu Identifier	string	lo.resultColumn.loCluId
Clu Code	string	lo.resultColumn.loCluCode
Lo Desc	string	lo.resultColumn.loDescPlain
Category	string	lo.resultColumn.categoryName

#### JPQL Implementation

```

NATIVE:SELECT
    lo.id ,
    clu.id ,
    officialIdentifier.cd ,
    lodesc.plain ,
    cat.name
FROM
    KSLU_CLU clu
JOIN
    KSLU_CLU_LO_RELTN jn
    ON
        jn.CLU_ID = clu.ID
JOIN
    KSLO_LO lo
    ON
        jn.LO_ID = lo.ID
JOIN
    KSLU_CLU_IDENT officialIdentifier
    ON
        clu.OFFIC_CLU_ID = officialIdentifier.ID
JOIN
    KSLO_RICH_TEXT_T lodesc
    ON
        lodesc.ID = lo.RT_DESCR_ID
JOIN
    KSLO_LO_JN_LOCATEGORY jncat
    ON
        lo.id = jncat.lo_id
JOIN
    KSLO_LO_CATEGORY cat
    ON
        cat.id = jncat.locategory_id

```

## LO Search for all LOs by matching category (lo.search.loByCluCategory)

<b>Search Type Key</b>	lo.search.loByCluCategory
<b>Name</b>	LO Search for all LOs by matching category
<b>Description</b>	Returns all matching LOs ids

### 1 Possible Search Criteria

<b>Type Key</b>	lo.criteria.byCluCode
<b>Name</b>	LoByCategoryCriteria
<b>Description</b>	null

Name	Optional	DataType	Read Only	Type Key	Implementation

Course Code	true	string	true	lo.queryParam.cluOfficialIdentifier.cluCode	officialId entifier.c d
-------------	------	--------	------	---------------------------------------------	-------------------------------

### 5 Result Columns Returned

Type Key	lo.result.loClu
Name	Los and Related Clus
Description	Lo descs ids and Clu ids and Codes

Name	DataType	Type Key
Lo Identifier	string	lo.resultColumn.loId
Clu Identifier	string	lo.resultColumn.loCluId
Clu Code	string	lo.resultColumn.loCluCode
Lo Desc	string	lo.resultColumn.loDescPlain
Category	string	lo.resultColumn.categoryName

### JPQL Implementation

```
null
```

### LO Search for all categories (lo.search.loCategories)

Search Type Key	lo.search.loCategories
Name	LO Search for all categories
Description	Returns all matching category names

### 3 Possible Search Criteria

Type Key	lo.criteria.byOptionalCategoryName
Name	LoByCategoryCriteria
Description	null

Name	Optional	DataType	Read Only	Type Key	Implementation
Learning Objective Category Name	true	string	false	lo.queryParam.loOptionalCategoryName	cat.name

Plain Text	true	string	true	lo.queryParam.loDescPlain	lo.de scr.p lain
Group Categories	true	String	false	lo.queryParam.grou pCategories	null

## 6 Result Columns Returned

Type Key	lo.result.categoryName
Name	Los and Related Clus
Description	Lo descs ids and Clu ids and Codes

Name	DataType	Type Key
Category ID	string	lo.resultColumn.categoryId
Category	string	lo.resultColumn.categoryName
Category	string	lo.resultColumn.categoryType
Category	string	lo.resultColumn.categoryTypeName
Category	string	lo.resultColumn.categoryNameAndType
Category state	string	lo.resultColumn.categoryState

## JPQL Implementation

```
SELECT cat.id, cat.name, cat.loCategoryType.id, cat.loCategoryType.name, cat.name || ' - '
  || cat.loCategoryType.name, cat.state FROM LoCategory cat
```

## LO Search for all categories based on name, type and state (lo.search.loCategoriesByNameRepoTypeState)

Search Type Key	lo.search.loCategoriesByNameRepoTypeState
Name	LO Search for all categories based on name, type and state
Description	Returns all matching category ids

## 4 Possible Search Criteria

Type Key	lo.criteria.byCategoryNameRepoTypeState				
Name	LoByCategoryCriteria				
Description	null				
Name	Optional	DataType	Read Only	Type Key	Implementation

Learning Objective Category Name	false	string	false	lo.queryParam.loCategoryName	null
Learning Objective Repository Key	false	string	false	lo.queryParam.loCategoryRepo	null
Learning Objective Category Type	false	string	false	lo.queryParam.loCategoryType	null
Learning Objective Category State	false	string	false	lo.queryParam.loCategoryState	null

## 1 Result Columns Returned

Type Key	lo.result.categoryId	
Name	LoCategory id	
Description	LoCategory id	
Name	DataType	Type Key
Category ID	string	lo.resultColumn.categoryId

## JPQL Implementation

```
SELECT cat.id FROM LoCategory cat WHERE LOWER(cat.name) =
:loQueryParam_loCategoryName AND cat.loRepository.id = :loQueryParam_loCategoryRepo
AND cat.loCategoryType.id = :loQueryParam_loCategoryType AND cat.state =
:loQueryParam_loCategoryState
```

## (CM 2.0) Formatted View of LRC Searches

 This page was automatically generated on Tue Dec 11 12:23:41 EST 2012  
DO NOT UPDATE MANUALLY!

This page represents a formatted view of lrc-search-config.xml

- Basic Search for Result Components (lrc.search.resultComponent)
  - 4 Possible Search Criteria
  - 3 Result Columns Returned
  - JPQL Implementation
- Basic Search for Result Values (lrc.search.resultValue)
  - 3 Possible Search Criteria
  - 2 Result Columns Returned
  - JPQL Implementation

## Basic Search for Result Components (lrc.search.resultComponent)

Search Type Key	lrc.search.resultComponent
Name	Basic Search for Result Components

<b>Description</b>	Return list of Result Components
--------------------	----------------------------------

#### 4 Possible Search Criteria

<b>Type Key</b>	Irc.criteria.resultComponent
<b>Name</b>	LuAdvancedCriteria
<b>Description</b>	Advanced criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
ResultComponent Key	true	string	true	IrcQueryParam.resultComponent.id	rc.id = :lrc_queryParam_resultComponent.id
ResultComponent Type Key	true	string	true	IrcQueryParam.resultComponent.type	rc.type IN (:lrc_queryParam_resultComponent.type)
Result Component Ids Key	true	list	true	IrcQueryParam.resultComponent.idRestrictionList	rc.id IN (:lrc_queryParam_resultComponent.idRestrictionList)

Result Component Ids Key	true	list	true	lrc.queryParam.resultComponent.resultscaleId	rc.resultscaleId IN (:lrc_queryParam_resultscaleId)
--------------------------	------	------	------	----------------------------------------------	-----------------------------------------------------

### 3 Result Columns Returned

Type Key	lrc.result.resultComponent	
Name	LRC Full View	
Description	Full view of ResultComponents	

Name	DataType	Type Key
ResultComponent Key	string	lrc.resultColumn.resultComponent.id
ResultComponent Name	string	lrc.resultColumn.resultComponent.name
ResultComponent Type	string	lrc.resultColumn.resultComponent.type

### JPQL Implementation

```
SELECT rc.id, rc.name, rc.type FROM ResultValuesGroupEntity rc
```

### Basic Search for Result Values (lrc.search.resultValue)

Search Type Key	lrc.search.resultValue	
Name	Basic Search for Result Values	
Description	Return list of Result Values	

### 3 Possible Search Criteria

Type Key	lrc.criteria.resultValue				
Name	Result Value Search Criteria				
Description	Search Criteria for Result Values such as actual grades				
Name	Optional	DataType	Read Only	Type Key	Implementation

Result Value Key	true	string	true	lrc.queryParam.resultValue.id	null
Result Value Key	true	string	true	lrc.queryParam.resultValue.value	null
Result Value's Associated Result Component Id	true	string	true	lrc.queryParam.resultValue.resultComponent.id	null

## 2 Result Columns Returned

Type Key	lrc.result.resultValue	
Name	Learning Result Values	
Description	Learning Result Values	

Name	DataType	Type Key
Result Value Key	string	lrc.resultColumn.resultValue.id
The Result Value	string	lrc.resultColumn.resultValue.value

## JPQL Implementation

null

## (CM 2.0) Formatted View of Organization Searches

⚠ This page was automatically generated on Tue Dec 11 12:23:33 EST 2012  
DO NOT UPDATE MANUALLY!

This page represents a formatted view of [organization-search-config.xml](#)

- Org Quick View by Relation Type, Org Type and Related Org Ids ([org.search.orgQuickViewByRelationTypeOrgTypeRelatedOrgIds](#))
  - 4 Possible Search Criteria
  - 4 Result Columns Returned
  - JPQL Implementation
- Generic search for organizations ([org.search.generic](#))
  - 7 Possible Search Criteria
  - 4 Result Columns Returned
  - JPQL Implementation
- All Orgs Filtered ([org.search.test.orgs](#))
  - 2 Possible Search Criteria
  - 5 Result Columns Returned
  - JPQL Implementation
- All org Hierarchies ([org.search.orgHierarchyIds](#))
  - 0 Possible Search Criteria
  - 2 Result Columns Returned
  - JPQL Implementation
- All org Types ([org.search.orgTypes](#))
  - 1 Possible Search Criteria
  - 2 Result Columns Returned
  - JPQL Implementation
- All org Types ([org.search.orgPersonRelationTypes](#))
  - 1 Possible Search Criteria
  - 2 Result Columns Returned

- JPQL Implementation
  - All org Hierarchies ([org.search.findOrgPositionRestrictions](#))
    - 1 Possible Search Criteria
    - 2 Result Columns Returned
    - JPQL Implementation
  - All org Hierarchies ([org.search.hierarchiesOrgIsIn](#))
    - 1 Possible Search Criteria
    - 1 Result Columns Returned
    - JPQL Implementation
  - Organization Quick View By Org Type ([org.search.orgQuickViewByOrgType](#))
    - 1 Possible Search Criteria
    - 2 Result Columns Returned
    - JPQL Implementation
  - Organization Quick View By Hierarchy and ShortName match ([org.search.orgQuickViewByHierarchyShortName](#))
    - 2 Possible Search Criteria
    - 2 Result Columns Returned
    - JPQL Implementation
  - Organization Quick View By Relation Type, Org Type and RelatedOrg type ([org.search.orgQuickViewByRelationTypeOrgTypeRelatedOrgType](#))
    - 3 Possible Search Criteria
    - 3 Result Columns Returned
    - JPQL Implementation
  - Organization Quick View By Relation Type, Org Type and RelatedOrg type ([org.search.orgQuickViewByRelationTypeOrgTypeRelatedOrgTypeAltr](#))
    - 3 Possible Search Criteria
    - 2 Result Columns Returned
    - JPQL Implementation
  - Organization Quick View By Relation Type, Org Type and Org name ([org.search.orgByRelationTypeOrgTypeOrgName](#))
    - 4 Possible Search Criteria
    - 4 Result Columns Returned
    - JPQL Implementation
  - Organization Quick View By Relation Type, Org Type or Org name ([org.search.orgByRelationTypeOrgType](#))
    - 6 Possible Search Criteria
    - 4 Result Columns Returned
    - JPQL Implementation
  - Organization Quick View By Relation Type, Org Ide and RelatedOrg type ([org.search.orgQuickViewByRelationTypeRelatedOrgTypeOrgId](#))
    - 3 Possible Search Criteria
    - 2 Result Columns Returned
    - JPQL Implementation
  - ShortName match ([org.search.orgByShortName](#))
    - 2 Possible Search Criteria
    - 2 Result Columns Returned
    - JPQL Implementation
  - ShortName and Type match ([org.search.orgByShortNameAndType](#))
    - 3 Possible Search Criteria
    - 2 Result Columns Returned
    - JPQL Implementation
  - Organization Quick View by First Letter of Short Name or Long Name ([org.search.orgQuickLongViewByFirstLetter](#))
    - 1 Possible Search Criteria
    - 4 Result Columns Returned
    - JPQL Implementation
  - Full view by organization id key ([org.search.orgFullViewById](#))
    - 1 Possible Search Criteria
    - 5 Result Columns Returned
    - JPQL Implementation
- 

## Org Quick View by Relation Type, Org Type and Related Org Ids ([org.search.orgQuickViewByRelationTypeOrgTypeRelatedOrgIds](#))

<b>Search Type Key</b>	<code>org.search.orgQuickViewByRelationTypeOrgTypeRelatedOrgIds</code>
<b>Name</b>	Org Quick View by Relation Type, Org Type and Related Org Ids
<b>Description</b>	Finding parent orgs, given the child org ids, parent relation type and parent org type

### 4 Possible Search Criteria

Type Key	org.criteria.orgQuickViewByRelationTypeOrgTypeRelatedOrgIds				
Name	OrgByRelationTypeOrgTypeRelatedOrgIds				
Description	child org ids, parent relation type and parent org type				
Name	Optional	DataType	Read Only	Type Key	Implementation
type	true	String	false	org.queryParam.optionalOrgTypeList	rel.org.type.id IN (:org_quer_yParam_opt_ional_OrgTypeList)

lds	true	list	false	org.queryParam.relatedOrgIds	(rel. relatedOrg .id IN (:org _queryPara m_rel atedO rgIds ) OR (rel. relatedOrg .id NOT IN (:org _queryPara m_rel atedO rgIds ) AND rel2. relatedOrg .id IN (:org _queryPara m_rel atedO rgIds ) AND rel.r elate dOrg. id = rel2. org.i d))
-----	------	------	-------	------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

type	true	String	false	org.queryParam.optionalRelationType	rel.type.id = :org_queryParam_optionalRelationType AND rel2.type.id = :org_queryParam_optionalRelationType
id	true	String	false	org.queryParam.relOrgOptionalId	rel.org.id = :org_queryParam_relOrgOptionalId

#### 4 Result Columns Returned

Type Key	org.result.generic
Name	Organization Full View Less Hierarchies
Description	Full view of the Organization Less Hierarchies

Name	DataType	Type Key
Organization Identifier	string	org.resultColumn.orgId
Organization Short Name	string	org.resultColumn.orgShortName
Name	string	org.resultColumn.orgOptionalLongName
Organization Type	string	org.resultColumn.orgType

#### JPQL Implementation

```
SELECT DISTINCT rel.org.id, rel.org.shortName, rel.org.longName, rel.org.type.id FROM
OrgOrgRelation rel, OrgOrgRelation rel2
```

## Generic search for organizations (org.search.generic)

<b>Search Type Key</b>	org.search.generic
<b>Name</b>	Generic search for organizations
<b>Description</b>	Query with multiple optional elements to satisfy most advanced pickers

### 7 Possible Search Criteria

<b>Type Key</b>	org.criteria.generic
<b>Name</b>	List of generic criteria
<b>Description</b>	Generic criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
longName	true	String	false	org.queryParam.org OptionalLongName	org.l ongNa me
Location	true	string	true	org.queryParam.org OptionalLocation	org.l ongNa me
id	true	String	false	org.queryParam.org OptionalId	org.i d = :org_ query Param _orgO ption alId

ids	true	list	false	org.queryParam.org OptionalIds	org.i d IN (:org _quer yPara m_org Optio nalId s)
shortName	true	String	false	org.queryParam.org OptionalShortName	org.s hortN ame
shortName	true	String	false	org.queryParam.star tswith.orgOptionalSh ortName	org.s hortN ame like :org_ query Param _star tswit h_org Optio nalSh ortNa me    '%' OR org.s hortN ame like '%'    :org_ query Param _star tswit h_org Optio nalSh ortNa me    '%'

type	true	String	false	org.queryParam.org OptionalType	org.type.id IN (:org_quer_yParam_org_OptionalType)
------	------	--------	-------	------------------------------------	----------------------------------------------------

#### 4 Result Columns Returned

Type Key	org.result.generic
Name	Organization Full View Less Hierarchies
Description	Full view of the Organization Less Hierarchies

Name	DataType	Type Key
Organization Identifier	string	org.resultColumn.orgId
Organization Short Name	string	org.resultColumn.orgShortName
Name	string	org.resultColumn.orgOptionalLongName
Organization Type	string	org.resultColumn.orgType

#### JPQL Implementation

```
SELECT org.id, org.shortName, org.longName, org.type.id FROM Org org
```

#### All Orgs Filtered (org.search.test.orgs)

Search Type Key	org.search.test.orgs
Name	All Orgs Filtered
Description	Query with multiple optional elements

#### 2 Possible Search Criteria

Type Key	org.criteria.orgGenericCriteria
Name	OrgGenericCriteria
Description	OrgGenericCriteria Description

Name	Optional	DataType	Read Only	Type Key	Implementation
------	----------	----------	-----------	----------	----------------

type	true	String	false	org.queryParam.orgGenericType	org.type.id = :org_queryParam_orgGenericType
shortName	true	String	false	org.queryParam.orgGenericShortName	org.shortName like :org_queryParam_orgGenericShortName

## 5 Result Columns Returned

Type Key	org.result.orgFullView
Name	Organization Full View
Description	Full view of the Organization

Name	DataType	Type Key
Organization Identifier	string	org.resultColumn.orgId
Organization Short Name	string	org.resultColumn.orgShortName
Organization Long Name	string	org.resultColumn.orgLongName
Organization Type	string	org.resultColumn.orgType
Organization Hierarchy Name	string	org.resultColumn.orgHierarchyName

## JPQL Implementation

```
SELECT org.id, org.shortName, org.longName, org.type.id, hierarchy.name FROM Org org,
in(org.type.orgHierarchies) hierarchy
```

## All org Hierarchies (org.search.orgHierarchyIds)

Search Type Key	org.search.orgHierarchyIds
Name	All org Hierarchies

Description	Returns all org hierarchy ids.
-------------	--------------------------------

#### 0 Possible Search Criteria

Type Key	org.criteria.orgHierarchyIds
Name	OrgHierarchyIdsCriteria
Description	OrgHierarchyIdsCriteria Description

Name	Optional	DataType	Read Only	Type Key	Implementation
------	----------	----------	-----------	----------	----------------

#### 2 Result Columns Returned

Type Key	org.result.orgHierarchyIds
Name	Organization Quick View
Description	Quick view of the Organization

Name	DataType	Type Key
Organization Hierarchy Id	string	org.resultColumn.orgHierarchyId
Organization Hierarchy Name	string	org.resultColumn.orgHierarchyName

#### JPQL Implementation

```
SELECT oh.id, oh.name FROM OrgHierarchy oh
```

#### All org Types (org.search.orgTypes)

Search Type Key	org.search.orgTypes
Name	All org Types
Description	Returns all org types.

#### 1 Possible Search Criteria

Type Key	org.criteria.orgType
Name	OrgTypeCriteria
Description	OrgTypeCriteria Description

Name	Optional	DataType	Read Only	Type Key	Implementation
id	true	String	false	org.queryParam.orgOptionalId	org.i d = :org_ query Param _orgO ption alId

## 2 Result Columns Returned

Type Key	org.result.orgType
Name	Organization Type
Description	Organization Type

Name	DataType	Type Key
Organization Type	string	org.resultColumn.orgType
Organization Type Name	string	org.resultColumn.orgTypeName

## JPQL Implementation

```
SELECT org.id, org.name FROM OrgType org
```

## All org Types (org.search.orgPeronRelationTypes)

Search Type Key	org.search.orgPeronRelationTypes
Name	All org Types
Description	Returns all org types.

## 1 Possible Search Criteria

Type Key	org.criteria.orgPersonRelationType
Name	OrgPersonRelationTypeCriteria
Description	OrgPersonRelationTypeCriteria Description

Name	Optional	DataType	Read Only	Type Key	Implementation
id	true	String	false	org.queryParam.orgOptionalId	org.i d = :org_ query Param _orgO ption alId

## 2 Result Columns Returned

Type Key	org.result.orgPersonRelationType
Name	Organization Person RelationType
Description	Organization Person RelationType

Name	DataType	Type Key
Organization Type	string	org.resultColumn.orgPersonRelationType

Organization Type Name	string	org.resultColumn.orgPersonRelationTypeNa me
------------------------	--------	------------------------------------------------

### JPQL Implementation

```
SELECT org.id, org.name FROM OrgPersonRelationType org
```

### All org Hierarchies (org.search.findOrgPositionRestrictions)

Search Type Key	org.search.findOrgPositionRestrictions
Name	All org Hierarchies
Description	Returns all org hierarchy ids.

### 1 Possible Search Criteria

Type Key	org.criteria.findOrgPositionRestrictions				
Name	findOrgPositionRestrictionsCriteria				
Description	Org ID on which to search for Positions				
Name	Optional	DataType	Read Only	Type Key	Implementation
id	false	String	false	org.queryParam.orgId	null

### 2 Result Columns Returned

Type Key	org.result.findOrgPositionRestrictions	
Name	Organization Position Restriction	
Description	Organization Position Restriction	
Name	DataType	Type Key
Organization Type	string	org.resultColumn.orgPositionRestrictionType
Organization Type Name	string	org.resultColumn.orgPositionRestrictionType Name

### JPQL Implementation

```
SELECT opr.personRelationType.id , opr.title FROM OrgPositionRestriction opr WHERE  
opr.org.id = :org_queryParam_orgId
```

### All org Hierarchies (org.search.hierarchiesOrgIsIn)

Search Type Key	org.search.hierarchiesOrgIsIn
Name	All org Hierarchies

<b>Description</b>	Returns all org hierarchy ids.
--------------------	--------------------------------

### 1 Possible Search Criteria

<b>Type Key</b>	org.criteria.hierarchiesOrgIsIn
<b>Name</b>	HierarchiesOrgIsInCriteria
<b>Description</b>	Org ID on which to search for containing hierarchies

Name	Optional	DataType	Read Only	Type Key	Implementation
id	false	String	false	orgQueryParam.orgId	null

### 1 Result Columns Returned

<b>Type Key</b>	org.result.hierarchiesOrgIsIn
<b>Name</b>	Hierarchies Containing an Org
<b>Description</b>	IDs of hierarchies that a specified org is a member of

Name	DataType	Type Key
Organization Hierarchy Id	string	org.resultColumn.orgHierarchyId

### JPQL Implementation

```
SELECT hierarchy.id FROM Org org, in(org.type.orgHierarchies) hierarchy WHERE org.id = :orgQueryParam_orgId
```

## Organization Quick View By Org Type (org.search.orgQuickViewByOrgType)

<b>Search Type Key</b>	org.search.orgQuickViewByOrgType
<b>Name</b>	Organization Quick View By Org Type
<b>Description</b>	Returns a quick view of organization information for orgs having a match on org type.

### 1 Possible Search Criteria

<b>Type Key</b>	org.criteria.orgByOrgType
<b>Name</b>	OrgByOrgTypeCriteria
<b>Description</b>	OrgByOrgTypeCriteria Description

Name	Optional	DataType	Read Only	Type Key	Implementation
type	false	String	false	orgQueryParam.orgType	null

### 2 Result Columns Returned

Type Key	org.result.orgQuickView
Name	Organization Quick View
Description	Quick view of the Organization

Name	DataType	Type Key
Organization Identifier	string	org.resultColumn.orgId
Organization Short Name	string	org.resultColumn.orgShortName

#### JPQL Implementation

```
SELECT org.id, org.shortName FROM Org org WHERE
    org.type.id = :orgQueryParam_orgType
```

#### Organization Quick View By Hierarchy and ShortName match (org.search.orgQuickViewByHierarchyShortName)

Search Type Key	org.search.orgQuickViewByHierarchyShortName
Name	Organization Quick View By Hierarchy and ShortName match
Description	Returns a quick view of organization information for orgs in a hierarchy with like on shortName.

#### 2 Possible Search Criteria

Type Key	org.criteria.orgByHierarchyShortName
Name	OrgByHierarchyShortNameCriteria
Description	OrgByHierarchyShortNameCriteria Description

Name	Optional	DataType	Read Only	Type Key	Implementation
orgHierarchyKey	false	String	false	org.queryParam.orgHierarchyId	null
shortName	false	String	false	org.queryParam.orgShortName	null

#### 2 Result Columns Returned

Type Key	org.result.orgQuickView
Name	Organization Quick View
Description	Quick view of the Organization

Name	DataType	Type Key
Organization Identifier	string	org.resultColumn.orgId
Organization Short Name	string	org.resultColumn.orgShortName

## JPQL Implementation

```
SELECT org.id, org.shortName FROM Org org, in(org.type.orgHierarchies) hierarchy WHERE
org.shortName like :org_queryParam_orgShortName and hierarchy.id =
:org_queryParam_orgHierarchyId
```

## Organization Quick View By Relation Type, Org Type and RelatedOrg type (org.search.orgQuickViewByRelationTypeOrgTypeRelatedOrgType)

<b>Search Type Key</b>	org.search.orgQuickViewByRelationTypeOrgTypeRelatedOrgType
<b>Name</b>	Organization Quick View By Relation Type, Org Type and RelatedOrg type
<b>Description</b>	Returns a quick view of organization information for orgs in a relation with specified types

### 3 Possible Search Criteria

<b>Type Key</b>	org.criteria.orgByRelationTypeOrgTypeRelatedOrgType
<b>Name</b>	OrgByRelationTypeOrgTypeRelatedOrgType
<b>Description</b>	Orgs in a relation with the given org type and related org type

Name	Optional	DataType	Read Only	Type Key	Implementation
type	false	String	false	org.queryParam.relationType	null
type	false	String	false	org.queryParam.orgType	null
type	false	String	false	org.queryParam.relatedOrgType	null

### 3 Result Columns Returned

<b>Type Key</b>	org.result.orgQuickViewExtended
<b>Name</b>	Organization Quick View
<b>Description</b>	Quick view of the Organization

Name	DataType	Type Key
Organization Identifier	string	org.resultColumn.orgId
Organization Short Name	string	org.resultColumn.orgShortName
Organization Long Name	string	org.resultColumn.orgLongName

## JPQL Implementation

```

SELECT rel.relatedOrg.id, rel.relatedOrg.shortName, rel.relatedOrg.longName FROM
OrgOrgRelation rel WHERE rel.type.id = :org_queryParam_relationType AND
rel.org.type.id = :org_queryParam_orgType AND rel.relatedOrg.type.id =
:org_queryParam_relatedOrgType ORDER BY rel.relatedOrg.shortName ASC

```

### Organization Quick View By Relation Type, Org Type and RelatedOrg type (org.search.orgQuickViewByRelationTypeOrgTypeRelatedOrgTypeAlt)

<b>Search Type Key</b>	org.search.orgQuickViewByRelationTypeOrgTypeRelatedOrgTypeAlt
<b>Name</b>	Organization Quick View By Relation Type, Org Type and RelatedOrg type
<b>Description</b>	Returns a quick view of organization information for orgs in a relation with specified types

### 3 Possible Search Criteria

<b>Type Key</b>	org.criteria.orgByRelationTypeOrgTypeRelatedOrgTypeAlt
<b>Name</b>	OrgByRelationTypeOrgTypeRelatedOrgType
<b>Description</b>	Orgs in a relation with the given org type and related org type

Name	Optional	DataType	Read Only	Type Key	Implementation
type	false	String	false	org.queryParam.relationType	null
type	false	String	false	org.queryParam.orgTypeList	null
type	false	String	false	org.queryParam.relatedOrgType	null

### 2 Result Columns Returned

<b>Type Key</b>	org.result.orgQuickView
<b>Name</b>	Organization Quick View
<b>Description</b>	Quick view of the Organization

Name	DataType	Type Key
Organization Identifier	string	org.resultColumn.orgId
Organization Short Name	string	org.resultColumn.orgShortName

### JPQL Implementation

```

SELECT rel.org.id, rel.org.shortName FROM OrgOrgRelation rel WHERE rel.type.id = :org_queryParam_relationType AND (rel.org.type.id in (:org_queryParam_orgTypeList)) AND rel.relatedOrg.type.id = :org_queryParam_relatedOrgType ORDER BY rel.org.shortName ASC

```

## Organization Quick View By Relation Type, Org Type and Org name (org.search.orgByRelationTypeOrgTypeOrgName)

<b>Search Type Key</b>	org.search.orgByRelationTypeOrgTypeOrgName
<b>Name</b>	Organization Quick View By Relation Type, Org Type and Org name
<b>Description</b>	Full view of organization information for orgs in a relation with specified types

### 4 Possible Search Criteria

<b>Type Key</b>	org.criteria.orgByRelationTypeOrgTypeOrgName
<b>Name</b>	OrgByRelationTypeOrgTypeOrgName
<b>Description</b>	Orgs in a relation with the given org type and org name

Name	Optional	DataType	Read Only	Type Key	Implementation
type	true	String	false	org.queryParam.optionalOrgRelationType	rel.type.id = :org_queryParam_relationType
type	true	String	false	org.queryParam.optionalOrgTypeList	rel.org.type.id IN (:org_queryParam_orgTypeList)

longName	true	String	false	org.queryParam.rel.OrgOptionalLongName	rel.org.longName
id	true	String	false	org.queryParam.orgOptionalId	org.id = :org_queryParam_orgOptionalId

#### 4 Result Columns Returned

Type Key	org.result.generic	
Name	Organization Full View Less Hierarchies	
Description	Full view of the Organization Less Hierarchies	

Name	DataType	Type Key
Organization Identifier	string	org.resultColumn.orgId
Organization Short Name	string	org.resultColumn.orgShortName
Name	string	org.resultColumn.orgOptionalLongName
Organization Type	string	org.resultColumn.orgType

#### JPQL Implementation

```
SELECT DISTINCT rel.org.id, rel.org.shortName, rel.org.longName, rel.org.type.id FROM OrgOrgRelation rel
```

#### Organization Quick View By Relation Type, Org Type or Org name (org.search.orgByRelationTypeOrgType)

Search Type Key	org.search.orgByRelationTypeOrgType
Name	Organization Quick View By Relation Type, Org Type or Org name
Description	Full view of organization information for orgs in a relation with specified types

#### 6 Possible Search Criteria

Type Key	org.criteria.orgByRelationTypeOrgType
Name	OrgByRelationTypeOrgType
Description	Orgs in a relation with the given org type

Name	Optional	DataType	Read Only	Type Key	Implementation
type	true	String	false	orgQueryParam.optionalOrgRelationType	rel.type.id = :org_queryParam_optionalOrgRelationType
type	true	String	false	orgQueryParam.optionalOrgTypeList	rel.org.type.id IN (:org_queryParam_optionalOrgTypeList)
longName	true	String	false	orgQueryParam.relOrgOptionalLongName	rel.org.longName
id	true	String	false	orgQueryParam.orgOptionalId	org.id = :org_queryParam_orgOptionalId
shortName	true	String	false	orgQueryParam.relOrgOptionalShortName	rel.org.shortName

Location	true	string	true	org.queryParam.orgOptionalLocation	org.locationName
----------	------	--------	------	------------------------------------	------------------

#### 4 Result Columns Returned

Type Key	org.result.generic
Name	Organization Full View Less Hierarchies
Description	Full view of the Organization Less Hierarchies

Name	DataType	Type Key
Organization Identifier	string	org.resultColumn.orgId
Organization Short Name	string	org.resultColumn.orgShortName
Name	string	org.resultColumn.orgOptionalLongName
Organization Type	string	org.resultColumn.orgType

#### JPQL Implementation

```
SELECT DISTINCT rel.org.id, rel.org.shortName, rel.org.longName, rel.org.type.id FROM
OrgOrgRelation rel
```

#### Organization Quick View By Relation Type, Org Ide and RelatedOrg type (org.search.orgQuickViewByRelationTypeRelatedOrgTypeOrgId)

Search Type Key	org.search.orgQuickViewByRelationTypeRelatedOrgTypeOrgId
Name	Organization Quick View By Relation Type, Org Ide and RelatedOrg type
Description	Returns a quick view of organization information for orgs in a relation with and org and specified types

#### 3 Possible Search Criteria

Type Key	org.criteria.orgQuickViewByRelationTypeRelatedOrgTypeOrgId
Name	OrgQuickViewByRelationTypeRelatedOrgTypeOrgId
Description	Orgs in a relation with the given orgid, type and related org name

Name	Optional	DataType	Read Only	Type Key	Implementation
type	false	String	false	org.queryParam.relationType	null
id	false	String	false	org.queryParam.orgId	null

type	false	String	false	org.queryParam.relatedOrgType	null
------	-------	--------	-------	-------------------------------	------

## 2 Result Columns Returned

Type Key	org.result.orgQuickView
Name	Organization Quick View
Description	Quick view of the Organization

Name	DataType	Type Key
Organization Identifier	string	org.resultColumn.orgId
Organization Short Name	string	org.resultColumn.orgShortName

## JPQL Implementation

```
SELECT rel.relatedOrg.id, rel.relatedOrg.shortName FROM OrgOrgRelation rel WHERE
rel.type.id = :orgQueryParam_relationType AND rel.org.id = :orgQueryParam_orgId AND
rel.relatedOrg.type.id = :orgQueryParam_relatedOrgType
```

## ShortName match (org.search.orgByShortName)

Search Type Key	org.search.orgByShortName
Name	ShortName match
Description	Search on shortName only.

## 2 Possible Search Criteria

Type Key	org.criteria.orgByShortName
Name	OrgByShortNameCriteria
Description	OrgByShortNameCriteria Description

Name	Optional	DataType	Read Only	Type Key	Implementation
shortName	false	String	false	org.queryParam.orgShortName	null
id	false	String	false	org.queryParam.orgId	null

## 2 Result Columns Returned

Type Key	org.result.orgQuickView
Name	Organization Quick View
Description	Quick view of the Organization

Name	DataType	Type Key
Organization Identifier	string	org.resultColumn.orgId
Organization Short Name	string	org.resultColumn.orgShortName

#### JPQL Implementation

```
SELECT org.id, org.shortName FROM Org org WHERE org.shortName like
:org_queryParam_orgShortName or org.id = :org_queryParam_orgId
```

#### ShortName and Type match (org.search.orgByShortNameAndType)

Search Type Key	org.search.orgByShortNameAndType
Name	ShortName and Type match
Description	Search on shortName and Type.

#### 3 Possible Search Criteria

Type Key	org.criteria.orgByShortNameAndType
Name	OrgByShortNameAndTypeCriteria
Description	OrgByShortNameAndTypeCriteria Description

Name	Optional	DataType	Read Only	Type Key	Implementation
shortName	false	String	false	org.queryParam.orgShortName	null
id	false	String	false	org.queryParam.orgId	null
type	false	String	false	org.queryParam.orgType	null

#### 2 Result Columns Returned

Type Key	org.result.orgQuickView
Name	Organization Quick View
Description	Quick view of the Organization

Name	DataType	Type Key
Organization Identifier	string	org.resultColumn.orgId
Organization Short Name	string	org.resultColumn.orgShortName

#### JPQL Implementation

```

SELECT org.id, org.shortName FROM Org org WHERE org.type.id = :orgQueryParam_orgType
AND (org.shortName like :orgQueryParam_orgShortName or org.id =
:orgQueryParam_orgId)

```

## Organization Quick View by First Letter of Short Name or Long Name (org.search.orgQuickLongViewByFirstLetter)

<b>Search Type Key</b>	org.search.orgQuickLongViewByFirstLetter
<b>Name</b>	Organization Quick View by First Letter of Short Name or Long Name
<b>Description</b>	Returns a quick view of organization information for orgs whose short or long name start with a particular letter

### 1 Possible Search Criteria

<b>Type Key</b>	org.criteria.orgByFirstLetter
<b>Name</b>	OrgByFirstLetterCriteria
<b>Description</b>	null

Name	Optional	DataType	Read Only	Type Key	Implementation
shortName	false	String	false	org.queryParam.orgShortName	null

### 4 Result Columns Returned

<b>Type Key</b>	org.result.orgQuickLongView
<b>Name</b>	Organization Quick View
<b>Description</b>	Quick view of the Organization

Name	DataType	Type Key
Organization Identifier	string	org.resultColumn.orgId
Organization Long Name	string	org.resultColumn.orgLongName
Organization Type	string	org.resultColumn.orgType
Organization Hierarchy Name	string	org.resultColumn.orgHierarchyName

### JPQL Implementation

```

SELECT org.id, org.longName, org.type.id, hierarchy.name FROM Org org,
in(org.type.orgHierarchies) hierarchy WHERE org.shortName like
:orgQueryParam_orgShortName or org.longName like :orgQueryParam_orgShortName

```

## Full view by organization id key (org.search.orgFullViewById)

<b>Search Type Key</b>	org.search.orgFullViewById
------------------------	----------------------------

<b>Name</b>	Full view by organization id key
<b>Description</b>	Returns a full view of organization information for the id supplied

## 1 Possible Search Criteria

<b>Type Key</b>	org.criteria.orgById
<b>Name</b>	OrgByIdCriteria
<b>Description</b>	OrgByIdCriteria Description

Name	Optional	DataType	Read Only	Type Key	Implementation
id	false	String	false	org.queryParam.orgId	null

## 5 Result Columns Returned

<b>Type Key</b>	org.result.orgFullView
<b>Name</b>	Organization Full View
<b>Description</b>	Full view of the Organization

Name	DataType	Type Key
Organization Identifier	string	org.resultColumn.orgId
Organization Short Name	string	org.resultColumn.orgShortName
Organization Long Name	string	org.resultColumn.orgLongName
Organization Type	string	org.resultColumn.orgType
Organization Hierarchy Name	string	org.resultColumn.orgHierarchyName

## JPQL Implementation

```
SELECT org.id, org.shortName, org.longName, org.type.id, hierarchy.name FROM Org org,
in(org.type.orgHierarchies) hierarchy WHERE org.id = :orgQueryParam_orgId
```

## (CM 2.0) Formatted View of Proposal Searches

! This page was automatically generated on Tue Dec 11 12:23:35 EST 2012  
DO NOT UPDATE MANUALLY!

This page represents a formatted view of [proposal-search-config.xml](#)

- Proposal search by course codes ([proposal.search.proposalsForReferencelds](#))
  - 1 Possible Search Criteria
  - 3 Result Columns Returned
  - JPQL Implementation
- Proposal search for clu proposals ([proposal.search.generic](#))
  - 3 Possible Search Criteria
  - 5 Result Columns Returned
  - JPQL Implementation
- Proposal search to return count of open proposals for a clu ([proposal.search.countForProposals](#))
  - 1 Possible Search Criteria
  - 1 Result Columns Returned
  - JPQL Implementation

## Proposal search by course codes (proposal.search.proposalsForReferencelds)

<b>Search Type Key</b>	proposal.search.proposalsForReferencelds
<b>Name</b>	Proposal search by course codes
<b>Description</b>	Returns all proposals that contain the course codes

### 1 Possible Search Criteria

<b>Type Key</b>	proposal.criteria.proposalsForReferencelds
<b>Name</b>	ProposalAdvancedCriteria
<b>Description</b>	Advanced criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
Proposal Name	true	string	false	proposal.queryParam.proposalOptionalReferencelds	null

### 3 Result Columns Returned

<b>Type Key</b>	proposal.result.proposalsForReferencelds
<b>Name</b>	Course Proposal Result
<b>Description</b>	List of course proposals for a given course code

Name	DataType	Type Key
Proposal Identifier	string	proposal.resultColumn.proposalId
Proposal Name	string	proposal.resultColumn.proposalOptionalName
Proposal Reference Id	string	proposal.resultColumn.proposalOptionalReferenceld

### JPQL Implementation

null

## Proposal search for clu proposals (proposal.search.generic)

<b>Search Type Key</b>	proposal.search.generic
<b>Name</b>	Proposal search for clu proposals
<b>Description</b>	Returns all proposals

### 3 Possible Search Criteria

<b>Type Key</b>	proposal.criteria.proposalAdvancedCriteria
<b>Name</b>	ProposalAdvancedCriteria
<b>Description</b>	Advanced criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
Proposal Identifier	true	string	true	proposal.queryParam.proposalId	null
Proposal Name	true	string	false	proposal.queryParam.proposalOptionalName	proposal.name
Proposal Name	true	string	false	proposal.queryParam.proposalOptionalType	proposal.ty pe.id IN(:p ropos al_qu eryPa ram_p ropos alOpt ional Type)

## 5 Result Columns Returned

Type Key	proposal.result.generic
Name	CourseProposalResult
Description	List of course proposals

Name	DataType	Type Key
Proposal Identifier	string	proposal.resultColumn.proposalId
Proposal Name	string	proposal.resultColumn.proposalOptionalName
Proposal Type	string	proposal.resultColumn.proposalOptionalType
Proposal Status	string	proposal.resultColumn.proposalOptionalStatus
Proposal Type name	string	proposal.resultColumn.proposalOptionalType

## JPQL Implementation

```
SELECT proposal.id, proposal.name, proposal.type.id, proposal.state, proposal.type.name
FROM Proposal proposal
```

## Proposal search to return count of open proposals for a clu (proposal.search.countForProposals)

<b>Search Type Key</b>	proposal.search.countForProposals
<b>Name</b>	Proposal search to return count of open proposals for a clu
<b>Description</b>	Returns count of open proposals

## 1 Possible Search Criteria

<b>Type Key</b>	proposal.criteria.countForProposalsCriteria
<b>Name</b>	CountForProposalsCriteria
<b>Description</b>	Count for proposals criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
Proposal Reference Type	false	string	false	proposal.queryParam.cluid	null

## 1 Result Columns Returned

<b>Type Key</b>	proposal.result.countForProposals
<b>Name</b>	ProposalCount
<b>Description</b>	Count of open proposals

Name	DataType	Type Key
Proposal Count	string	proposal.resultColumn.proposalCount

## JPQL Implementation

```
SELECT count (proposal.id) FROM Proposal proposal, IN (proposal.proposalReference) pr
where pr.objectReferenceId = :proposal_queryParam_cluId and
pr.type='kuali.proposal.referenceType.clu' and proposal.state IN ('Saved', 'Enroute')
```

## (CM 2.0) Formatted View of Statement Searches

⚠ This page was automatically generated on Tue Dec 11 12:23:35 EST 2012  
DO NOT UPDATE MANUALLY!

This page represents a formatted view of statement-search-config.xml

- Statement Dependency Analysis (stmt.search.dependencyAnalysis)
  - 2 Possible Search Criteria
  - 5 Result Columns Returned
  - JPQL Implementation
- Basic and Advanced Search (stmt.search.cluInReqComponent)
  - 2 Possible Search Criteria
  - 7 Result Columns Returned
  - JPQL Implementation

## Statement Dependency Analysis (stmt.search.dependencyAnalysis)

<b>Search Type Key</b>	stmt.search.dependencyAnalysis
<b>Name</b>	Statement Dependency Analysis

<b>Description</b>	Query Statements that reference the input clouds and clusters. Searches up the statement hierarchy
--------------------	-------------------------------------------------------------------------------------------------------

## 2 Possible Search Criteria

<b>Type Key</b>	stmt.criteria.dependencyAnalysis
<b>Name</b>	Requirement Component Type Relations
<b>Description</b>	Advanced criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
Requirement Component Field Type	false	string	false	stmt.queryParam.reqComponentFieldType	null
Requirement Component Field Value	false	string	false	stmt.queryParam.reqComponentFieldValue	null

## 5 Result Columns Returned

<b>Type Key</b>	stmt.result.dependencyAnalysis
<b>Name</b>	Statement Dependency Analysis Results
<b>Description</b>	Statement Dependency Analysis Results

Name	DataType	Type Key
Reference Object Id	string	stmt.resultColumn.refObjId
Root Id	string	stmt.resultColumn.rootId
Requirement Component Ids	string	stmt.resultColumn.requirementComponentIds
Statement Type Id	string	stmt.resultColumn.statementTypeId
Statement Type Name	string	stmt.resultColumn.statementTypeName

## JPQL Implementation

null
------

## Basic and Advanced Search (stmt.search.cluInReqComponent)

<b>Search Type Key</b>	stmt.search.cluInReqComponent
<b>Name</b>	Basic and Advanced Search
<b>Description</b>	Query with multiple optional elements to satisfy most advanced pickers

## 2 Possible Search Criteria

<b>Type Key</b>	stmt.criteria.reqComponentFieldType
-----------------	-------------------------------------

<b>Name</b>	Requirement Component Type Relations				
<b>Description</b>	Advanced criteria				

Name	Optional	DataType	Read Only	Type Key	Implementation
Requirement Component Field Type	false	string	false	stmt.queryParam.reqComponentFieldType	null
Requirement Component Field Value	false	string	false	stmt.queryParam.reqComponentFieldValue	null

## 7 Result Columns Returned

<b>Type Key</b>	stmt.result.clusInReqComponent	
<b>Name</b>	Lu Full View	
<b>Description</b>	Full view of clus	

Name	DataType	Type Key
Requirement Component Id	string	stmt.resultColumn.reqComponentId
Requirement Component Type Name	string	stmt.resultColumn.reqComponentTypeId
Requirement Component Type Name	string	stmt.resultColumn.reqComponentTypeName
Requirement Component Type Description	string	stmt.resultColumn.reqComponentTypeDesc
Statement Type Id	string	stmt.resultColumn.statementTypeId
Statement Type Name	string	stmt.resultColumn.statementTypeName
Statement Type Description	string	stmt.resultColumn.statementTypeDesc

## JPQL Implementation

```

SELECT rc.id, rc.requiredComponentType.id, rc.requiredComponentType.name,
rc.requiredComponentType.descr,
       stmtType.id, stmtType.name, stmtType.descr
FROM ReqComponent rc
JOIN rc.reqComponentFields reqComponentField
JOIN rc.requiredComponentType.statementTypes stmtType
WHERE reqComponentField.type = :stmt_queryParam_reqComponentFieldType
      AND reqComponentField.value = :stmt_queryParam_reqComponentFieldValue

```

## (CM 2.0) Formatted View of Subject Code Searches

⚠ This page was automatically generated on Tue Dec 11 12:23:35 EST 2012  
DO NOT UPDATE MANUALLY!

This page represents a formatted view of [subjectcode-search-config.xml](#)

- Generic Subject Code Search ([subjectCode.search.subjectCodeGeneric](#))
  - 1 Possible Search Criteria

- 4 Result Columns Returned
  - JPQL Implementation
  - Subject Code Organization Relations (subjectCode.search.orgsForSubjectCode)
    - 2 Possible Search Criteria
    - 8 Result Columns Returned
    - JPQL Implementation
- 

## Generic Subject Code Search (subjectCode.search.subjectCodeGeneric)

<b>Search Type Key</b>	subjectCode.search.subjectCodeGeneric
<b>Name</b>	Generic Subject Code Search
<b>Description</b>	Search for all Subject Codes

### 1 Possible Search Criteria

<b>Type Key</b>	subjectCode.criteria.generic
<b>Name</b>	Subject Code Criteria
<b>Description</b>	Subject Code Criteria

Name	Optional	DataType	Read Only	Type Key	Implementation
Subject Code	true	string	false	subjectCode.queryParam.code	null

### 4 Result Columns Returned

<b>Type Key</b>	subjectCode.results.generic
<b>Name</b>	Subject Code Results
<b>Description</b>	Subject Code Results

Name	DataType	Type Key
Subject Code Id	string	subjectCode.resultColumn.id
Subject Code	string	subjectCode.resultColumn.code
Subject Code Name	string	subjectCode.resultColumn.name
Subject Code Usage Type	string	subjectCode.resultColumn.type

### JPQL Implementation

```
null
```

## Subject Code Organization Relations (subjectCode.search.orgsForSubjectCode)

<b>Search Type Key</b>	subjectCode.search.orgsForSubjectCode
<b>Name</b>	Subject Code Organization Relations
<b>Description</b>	Search for all Organizations related to the given Subject Code

### 2 Possible Search Criteria

Type Key	subjectCode.criteria.orgsForSubjectCode				
Name	Subject Code Organizations Criteria				
Description	Subject Code Organizations Criteria				

Name	Optional	DataType	Read Only	Type Key	Implementation
Subject Code	true	string	false	subjectCode.queryParam.code	null
Organization Id	true	string	false	subjectCode.queryParam.optionalOrgId	null

#### 8 Result Columns Returned

Type Key	subjectCode.results.orgsForSubjectCode	
Name	Subject Code Results	
Description	Subject Code Results	

Name	DataType	Type Key
Subject Code	string	subjectCode.resultColumn.code
Subject Code Usage Type	string	subjectCode.resultColumn.type
Active From	date	subjectCode.resultColumn.activeFrom
Active To	date	subjectCode.resultColumn.activeTo
Org Id	string	subjectCode.resultColumn.orgId
Org Short Name	string	subjectCode.resultColumn.orgShortName
Org Long Name	string	subjectCode.resultColumn.orgLongName
Org Type	string	subjectCode.resultColumn.orgType

#### JPQL Implementation

```
null
```

## (CM 2.0) State Service

- Class
- Description
- Entity Diagram
- Service Contract Documentation
- Type and State Configuration

### Class

State Service is a Core service for state management.

### Description

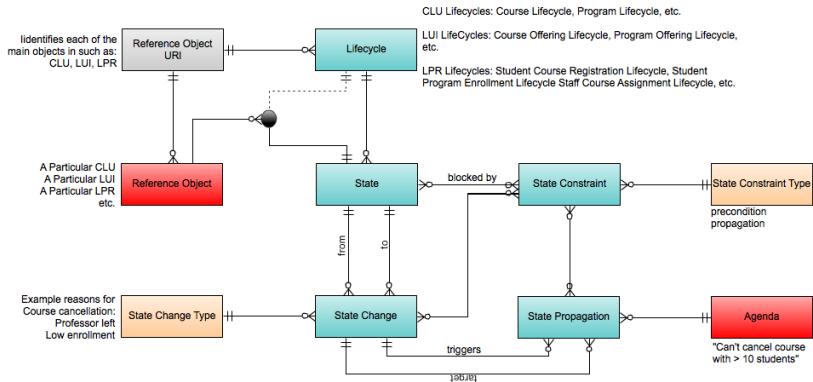
State service is used for managing the following: valid States of an object, constraints associated with a state change and propagation triggered

by a State change. It also describes a way to manage constraint and propagation of States.

More...

# Entity Diagram

## **State Service Entity Diagram**



More...

# Service Contract Documentation

[Go to StateService Contract Documentation](#)

## Type and State Configuration

More...

## **(CM 2.0) State Service Description and Assumptions**

## State Service

## Description

State service is used for managing the following: valid States of an object, constraints associated with a state change and propagation triggered by a State change. It also describes a way to manage constraint and propagation of States.

## Key Concepts

## State Event Features



- *State change* from the current object state to another state triggered by an *event/reason*, subject to desired constraints.
  - *State Propagation* – an object's state change could trigger state changes in other objects, subject to desired constraints.
  - *Cross Constraints* – an object's current state may constrain the valid states of another object, subject to desired constraints.

## Lifecycle

- State service organizes collections of States into a Lifecycle.
- A Lifecycle indicates the type of entity to which a set of States apply. For example, a LUI may go through a lifecycle for instantiation with States of "rolled over," "scheduled," and "published."
- The "instantiation lifecycle" is only applicable to LUIs although the service contracts accept any identifier.

## State Propagation and Constraints

- Events, implemented as the type of StateChange object, can be used to capture the reason for a state change. For example, 'Professor left' and 'Low enrollment' could be two different events leading to the same Course Canceled state.
- Allow transition from any state to any state, and use 'constraints' as needed. Constraint operators are:
  - ALL: all related objects must be in a State
  - EXISTS: at least one of the related objects must be in a State
  - NONE: none of the related objects must be in a State

## Assumptions

- State setup is part of configuration and not managed specifically through the service.
- Lifecycle key is unique within the managing service.
- An object of a given type can have one or more lifecycles associated with it.
- State keys can be reused between different lifecycles.
- An entity can only be in one state at a time, therefore it can only participate in one lifecycle at a time. It is possible for different entities of the same type to participate in different lifecycles. It is even possible that the same entity participate in different lifecycles at different times.. So that there may be a "creation/approval" lifecycles and then later a "retire" lifecycle.

## Deferred Design

- management of Lifecycle entities
- management of State-Lifecycle mappings

## (CM 2.0) State Service Implementation Notes

- Related Highlevel Design Specifications
  - M6
- Milestone Changes
- Overview

## Related Highlevel Design Specifications

### M6

TODO Drop link to State Service HLDS here

## Milestone Changes

Milestone	Description
M6	Complex constraints will be modelled using a KRMS Agenda but that is <b>out-of-scope</b> for this milestone.

## Overview

State changes can be constrained by the overall system state and can cause cascading effects to other entities and their states.

The State Service provides a mechanism to author these state change constraints and on state changed propagations.

The intent is that the logic of what should happen on a state change is encoded within this service and then used to evaluate if a proposed state change is allowable.

Constraints related to the state of an object or states on a related object can be handled directly in the service entity model. Anything more complex will be externalized through a KRMS Agenda. Essentially if the rules in the agenda evaluate to true then the state change is allowed to proceed.

KRMS related work is out of scope in M6.

## (CM 2.0) State Service FAQ

- What is a lifecycle key and how does it map to a state?
- Do we need an ordering on the next set of happy states?
- You say states can be modeled as a "finite state machine" but you don't provide all the methods to do that.

- How might I implement this partial finite state machine?

## What is a lifecycle key and how does it map to a state?

We decided to add "LIFECYCLE\_KEY" column in to table state, to map with this new column, we need to have lifecycle in StateInfo, right?

**Ans:**

Unfortunately it can't be as simple as that. A single state can (potentially) be connected to more than one lifecycle key.  
Although most of the time there is just one lifecycle key per object it is possible that there are multiple lifecycle keys that reference the same state.

TODO: get a good example for this.

## Do we need an ordering on the next set of happy states?

The service team defined the Next Happy State to support UI requirements to show what might be required at the next happy state on the path. If there are two paths that both seem "happy" then one of them must be chosen as the next most likely happy path.

## You say states can be modeled as a "finite state machine" but you don't provide all the methods to do that.

yeah... but we hesitated to actually implement a fully instantiated finite state machine because that is really workflow and gets really complicated and should really belong in the domain of KEW.

Instead we focussed just on the operations that are immediately needed to move forward. There were 2 questions in R1 that were always being asked that the service could not answer:

- (1) What value should we default the state of the object to (or what values are legal to set on the object when we first create it) and
- (2) What is the next happy state so we can calculate requiredness for that state (required to submit)

getInitialStates answers question (1) getNextHappyState answers question (2)

## How might I implement this partial finite state machine?

Lifecycle State Transition Table

Lifecycle key	PriorStateKey	NextStateKey
lifecycle 1	(none)	state a (initial)
lifecycle 1	state a	state b (next happy)
lifecycle 1	state b	state c (next happy)
lifecycle 1	(none)	state b (initial)

In this table:

1. initial states are rows where there is no "prior" state
2. Next happy states are those with a prior state key

## (CM 2.0) State service Types and States

## (CM 2.0) State Types and States

## State Change Types and states

StateChange Types

Type Key	Type Name	Type Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions

kuali.state.change.type	State Change	A State Change		Configured	Y	Y	<input checked="" type="checkbox"/> KSENR 904 - Create Types/States for Service as Placeholders for Team Review (Closed)
-------------------------	--------------	----------------	--	------------	---	---	-----------------------------------------------------------------------------------------------------------------------------

## StateChange States

Lifecycle Key
kuali.state.change.lifecycle

State Key	State Name	State Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.state.change.state.active (initial)	Active	A default state of active		Configured	Y	Y	<input checked="" type="checkbox"/> KSENR 904 - Create Types/States for Service as Placeholders for Team Review (Closed)

## State Constraint Types and states

### StateConstraint Types

Type Key	Type Name	Type Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.state.constraint.type.precondition	Precondition	Constraint for a state change		Configured	Y	Y	<input checked="" type="checkbox"/> KSENR 904 - Create Types/States for Service as Placeholders for Team Review (Closed)
kuali.state.constraint.type.propagation	Propagation	Constraint for a propagation		Configured	Y	Y	<input checked="" type="checkbox"/> KSENR 904 - Create Types/States for Service as Placeholders for Team Review (Closed)

### StateConstraint States

## Lifecycle Key

kuali.state.constraint.lifecycle

State Key	State Name	State Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.state.constraint.state.active (initial)	Active	A default state of active		Configured	Y	Y	<input checked="" type="checkbox"/> KSENR 904 - Create Types/States for Service as Placeholders for Team Review (Closed)

## State Propagation Types and states

### StatePropagation Types

Type Key	Type Name	StatePropagation Type Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.state.propagation.type	State Propagation	A State Propagation.		Configured	Y	Y	<input checked="" type="checkbox"/> KSENR 904 - Create Types/States for Service as Placeholders for Team Review (Closed)

### StatePropagation States

## Lifecycle Key

kuali.state.propagation.lifecycle

State Key	State Name	State Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.state.propagation.state.active (initial)	Active	A default state of active		Configured	Y	Y	<input checked="" type="checkbox"/> KSENR 904 - Create Types/States for Service as Placeholders for Team Review (Closed)

## (CM 2.0) State Change States

### StateChange States

## Lifecycle Key

kuali.state.change.lifecycle

State Key	State Name	State Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.state.change.state.active (initial)	Active	A default state of active		Configured	Y	Y	<input checked="" type="checkbox"/> KSENR 904 - Create Types/States for Service as Placeholders for Team Review (Closed)

## (CM 2.0) State Change Types

### StateChange Types

Type Key	Type Name	Type Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.state.change.type	State Change	A State Change		Configured	Y	Y	<input checked="" type="checkbox"/> KSENR 904 - Create Types/States for Service as Placeholders for Team Review (Closed)

## (CM 2.0) State Constraint States

### StateConstraint States

## Lifecycle Key

kuali.state.constraint.lifecycle

State Key	State Name	State Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Questions
kuali.state.constraint.state.active (initial)	Active	A default state of active		Configured	Y	Y	<input checked="" type="checkbox"/> KSENR 904 - Create Types/States for Service as Placeholders for Team Review (Closed)

## (CM 2.0) State Constraint Types

### StateConstraint Types

Type Key	Type Name	Type Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Q uestions
kuali.state.constraint.type.pre condition	Precondition	Constraint for a state change		Configured	Y	Y	<input checked="" type="checkbox"/> KSENR 904 - Create Types/States fo Service as Placeholders fo Team Review (- Closed)
kuali.state.constraint.type.propagation	Propagation	Constraint for a propagation		Configured	Y	Y	<input checked="" type="checkbox"/> KSENR 904 - Create Types/States fo Service as Placeholders fo Team Review (- Closed)

## (CM 2.0) State Propagation States

### *StatePropagation States*

Lifecycle Key
kuali.state.propagation.lifecycle

State Key	State Name	State Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Q uestions
kuali.state.propagation.state.active (initial)	Active	A default state of active		Configured	Y	Y	<input checked="" type="checkbox"/> KSENR 904 - Create Types/States fo Service as Placeholders fo Team Review (- Closed)

## (CM 2.0) State Propagation Types

### *StatePropagation Types*

Type Key	Type Name	StatePropagation Type Description	Aliases	Agreed Upon?	In Constant File?	In Database?	Comments/Q uestions
kuali.state.propagation.type	State Propagation	A State Propagation.		Configured	Y	Y	<input checked="" type="checkbox"/> KSENR 904 - Create Types/States fo Service as Placeholders fo Team Review (- Closed)

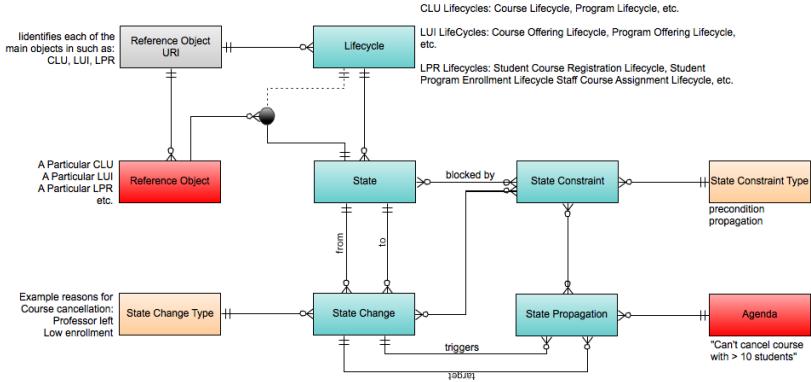
## (CM 2.0) State Service Entity Diagram

- State Service Entity Diagram

- Database Model

## State Service Entity Diagram

**State Service Entity Diagram**



## Database Model

[STUDENT:Database ER Diagram](#)

[STUDENT:Database Schema Create Script](#)

## (CM 2.0) Type Service

- Class
- Description
- Entity Diagram
- Service Contract Documentation
- Type and State Configuration

## Class

The Types Service is a Core service for managing Types.

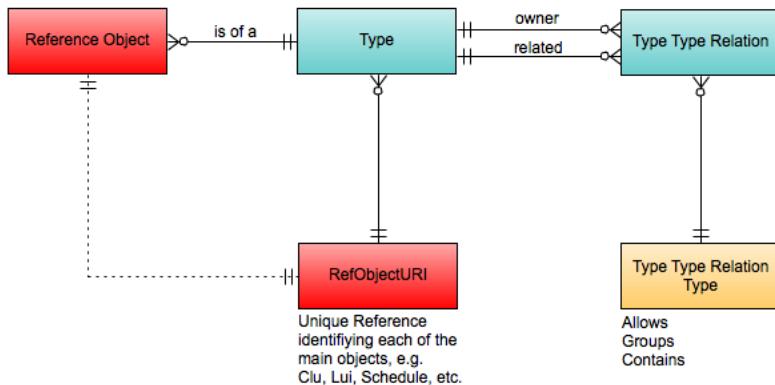
## Description

Type service is Kuali Student's generic way of reading type information. Type service exposes information about a specific type and also the relationship between types.

[More...](#)

## Entity Diagram

## Type Service Entity Diagram



More...

## Service Contract Documentation

[Go to TypeService Contract Documentation](#)

## Type and State Configuration

More...

## (CM 2.0) Type Service Description and Assumptions

### Type Service

#### Description

Type service is Kuali Student's generic way of reading type information. Type service exposes information about a specific type and also the relationship between types.

#### Key Concepts

Types can be grouped together within a service using the RefObjectURI. This enables defining and reading types that are applicable to a given Object in KS. e.g <http://student.kuali.org/LuService/CluInfo> will have types for Course, Formats, Activities etc

#### Type Type Relations

- Types are related to other types through a TypeTypeRelation entity. The default relationship is the 'kuali.relationtype.allowed' relationship between two Types. Valid relationships (captured through a type on the relationship) are
  1. kuali.relationtype.allowed
  2. kuali.relationtype.contains
  3. kuali.relationtype.group
- In case of a contains relation type, a rank can be provided to order the contained types.

OwnerTypeKey	RelatedTypeKey	Type-Type-Relation Type	Rank
kuali.statement.type.course.academicReadiness.prereq	course.courseset.completed.nof	kuali.relationtype.allowed	0

kuali.statement.type.course.academicReadiness.prereq	course.courseset.gpa.min	kuali.relationtype.allowed	0
kuali.atp.type.AY	kuali.atp.type.Fall	kuali.relationtype.contains	0
kuali.atp.type.AY	kuali.atp.type.Spring	kuali.relationtype.contains	1
kuali.atp.type.AY	kuali.atp.type.Summer	kuali.relationtype.contains	2
kuali.atp.type.Summer	kuali.atp.type.Mini-mesterA	kuali.relationtype.contains	0
kuali.atp.type.Summer	kuali.atp.type.Mini-mesterB	kuali.relationtype.contains	1
kuali.atp.type.Holiday	kuali.atp.type.SpringBreak	kuali.relationtype.group	0
kuali.atp.type.Holiday	kuali.atp.type.ThanksGiving	kuali.relationtype.group	0

#### Not All Types Are Tied To Objects

- Types that categorize or group other types
  - Duration Types are types that are used to categorize and describe an ATP type
  - Season Types ditto
  - Instructional Format types are used to categorize LU Types
  - Delivery Method Types are used to categorize LU Types
- Types that drive processing
  - [Waitlist types](#).
  - [Registration Ordering Types](#)
  - [Check Types](#)
- State Lifecycle Process Type Keys

#### Assumptions

- Type setup is part of configuration and not managed specifically through the service.
- Type Type Relation setup is part of configuration and not managed specifically through the service.
- Type key is unique within the implementing service.
- Type relationships will be read and enforced during the creation of the objects.

## (CM 2.0) TypeTypeRelation Types and States

### Types

The page TypeType Relation Types could not be found.

### States

The page TypeType Relations States could not be found.

## (CM 2.0) TypeType Relations States

### TypeType Relation States

State Key	Name	Description	Aliases	Agreed Upon?	Notes/Comments
kuali.type.type.relation.state.active	Active	Relation is Active		Proposed	
kuali.type.type.relation.state.inactive	Inactive	Relation is in-active		Proposed	

## (CM 2.0) TypeType Relation Types

### TypeTypeRelation Types

Type Key	Name	Description	Aliases	Examples	Agreed Upon?	Notes/Comments

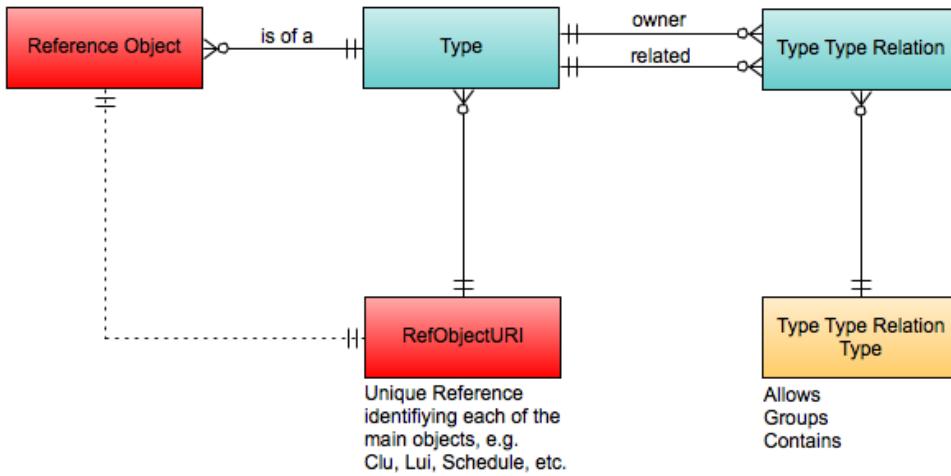
kuali.type.type.relation.type.allowed	Allowed Type Type Relation	Type specifies which types can go together	Constraint Propagation	Half Semester terms can only be sub-terms of the Fall or Spring semesters.	Proposed	
kuali.type.type.relation.type.contains	Contains Type Type Relation	Type that are contained within another type		???	Placeholder	No real use case yet
kuali.type.type.relation.type.group	Group Type Type Relation	Types grouped together	Collected, Categorized	Milestone types are grouped by whether they are holidays, events or key dates and then within holidays whether they are instructional or not.  ATP types are grouped into calendars and terms.  ATPs are also grouped as to their seasonality (fall, spring, etc) and duration (semester, quarter)	Proposed	

## (CM 2.0) Type Service Entity Diagram

- Type Service Entity Diagram
- Type Service Database Model

### Type Service Entity Diagram

## Type Service Entity Diagram



## Type Service Database Model

[STUDENT:Database ER Diagram](#)  
[STUDENT:Database Create Script](#)

## (CM 2.0) Version Management Service

- Class
- Description
- Entity Diagram
- Service Contract Documentation

### Class

Version Management is a Core service that supports versioning of objects.

### Description

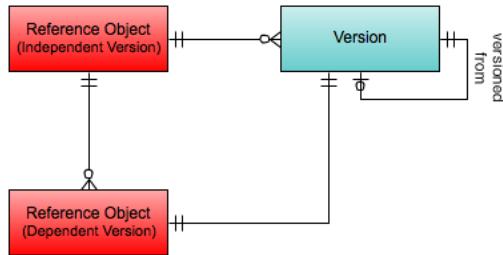
The Version Management Service inspects versions of an object. The Version Management Service is used in conjunction with another service to examine the version history of an object. This service provides access to the identifiers to retrieve older or future versions of objects from their respective services.

The Version Management Service is only useful when used in conjunction with another service that supports versioning.

More...

### Entity Diagram

## Version Management Service Entity Diagram



## Service Contract Documentation

[Go to Version Management Service Contract Documentation](#)

## (CM 2.0) Version Management Service Description and Assumptions

### Version Management Service

#### Description

The Version Management Service inspects versions of an object. The Version Management Service is used in conjunction with another service to examine the version history of an object. This service provides access to the identifiers to retrieve older or future versions of objects from their respective services.

The Version Management Service is only useful when used in conjunction with another service that supports versioning.

#### Key Concepts

##### Terminology

- Versioned object: an object at a specific version.
- Current object: an object that is at the current version. The current version may not be the latest version.
- Draft object: an object that is at a higher version level than the active object.
- Version Specific Id: an identifier that identifies a particular version of an object.
- Version Independent Id: an identifier that always refers to the object at the current version. It is independent of the version.

##### Id Management

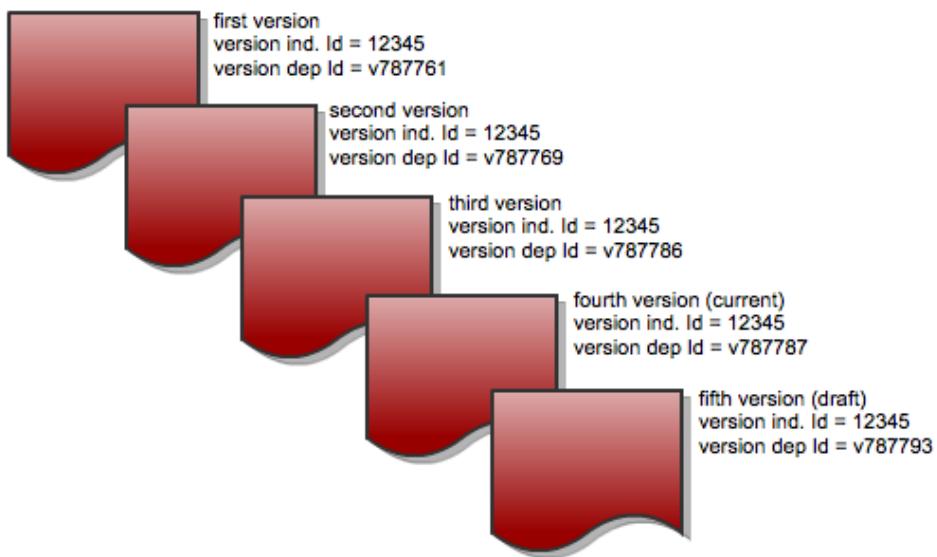
Typically, every object is identified by a unique Id. A service that supports version management identifies each version of an object by its own unique Id. This is the version specific Id. A version specific identifier is assigned to an object for each new version created.

When a version specific identifier is used within a service that supports version management, the operations and relationships refer to a specific version of that object. When new versions of the object are created, the version specific identifier continues to refer to the older version.

A single version of an object is designated as the current version. The current version of an object is identified by a version independent identifier. The version independent identifier is assigned when the first version of an object is created and never changes as new versions of an object are created.

When a version independent identifier is used within a service that supports version management, the operations and relationships refer to the current version of that object. When the current version of an object moves forward to the next version, the operations and relationships refer to the next (now current) version.

## Version Chain



## Versions

- The version history of an object is retrieved using lookup methods in the Version Management service using the version specific or independent identifier of the reference object.
- The VersionDisplay object is used in the Version Management Service to examine the versioning information of an object.

## Example

```
// get the version history of a CLU
VersionDisplayInfoList versions = VersionManagementService.getVersions(CLUREF_URI,
clu.getId(), context);

// get the first version of a CLU
VersionDisplayInfo version = VersionManagementService.getFirstVersion(CLUREF_URI,
clu.getId(), context);
Clu firstClu = CluService.getClu(version.getId(), context);

// gets the current version of a CLU
VersionDisplayInfo version = VersionManagementService.getCurrentVersion(CLUREF_URI,
clu.getId(), context);
Clu currentClu = CluService.getClu(version.getId(), context);
```

## Compatibility Issues

Some consumers are "version aware" and other consumers are "version unaware." To separate these two different use cases, a service supporting version management must be able to distinguish version specific Ids from version independent Ids.

- `getClu(versionIndependentId)` returns the current version of the Clu. `clu.getId()` returns the version independent Id.
- `getClu(versionSpecificId)` returns a specific version of the Clu (which may coincidentally be the current version). `clu.getId()` in this case returns the version specific Id.

When versioning support is added to a service, version unaware applications continue to use the same service and never risk getting their hands on version specific identifiers. Version aware applications express their ability to manage multiple versions of the same object by retrieving a version specific identifier from the Version Management Service.

## Issues

- This service does not offer operations to create versions, only inspect them. Currently, the creation of new versions explicitly occurs in other services.
- This service does not support branching. There is only one next version for any version.
- There are no Version Types, which might be useful for examining different kinds of versions.
- Embedding the Version inside each object.
- Implementations need to distinguish Ids among different classes of objects so we don't need to specify URIs as a separate argument.

## (CM 2.0) Version Management Service Entity Diagram

### Version Management Service Entity Diagram

