Kuali Rice 2.3.3 KEN Guide

Table of Contents

List of Figures

List of Tables

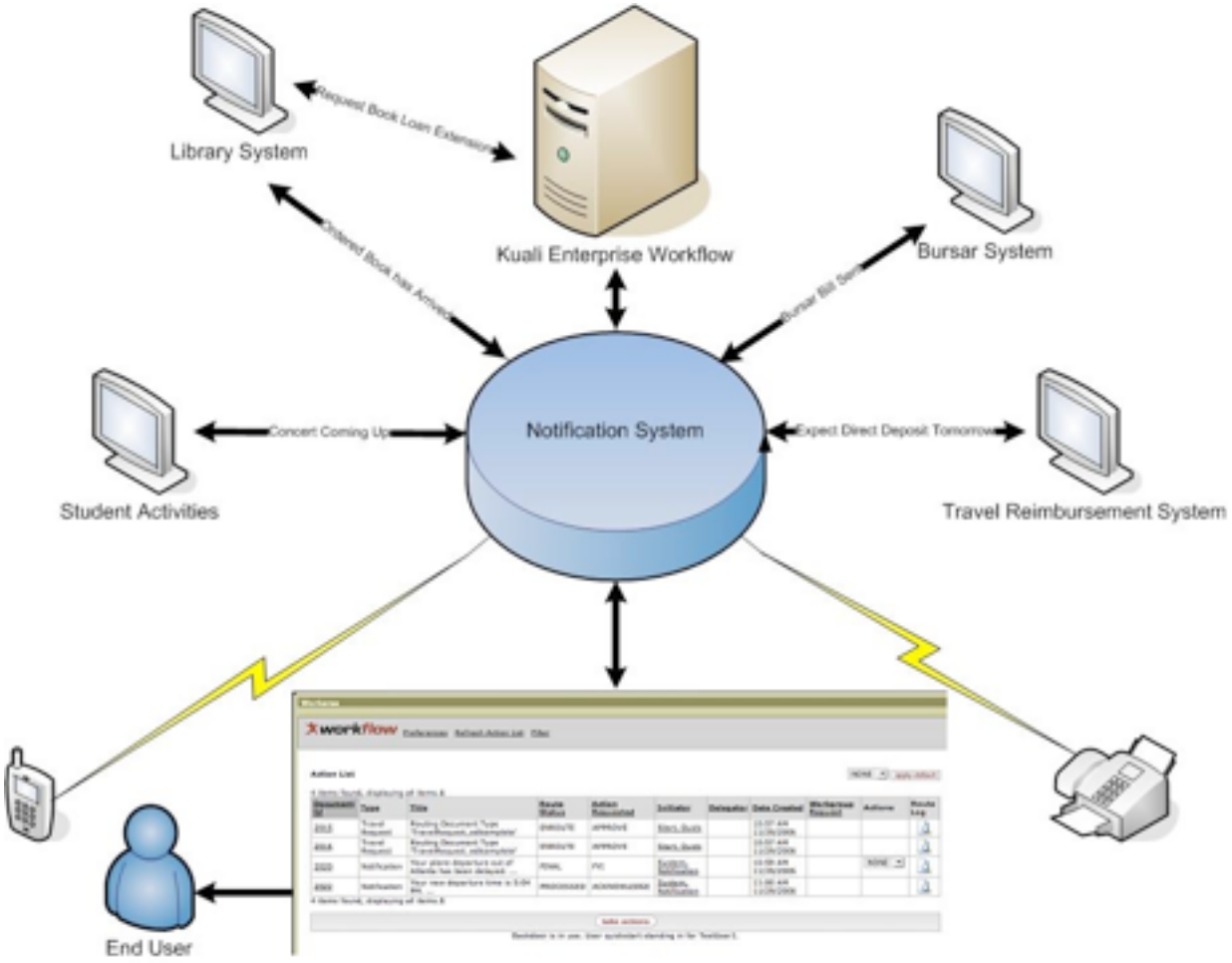
List of Examples

Chapter 1. KEN Overview

What is KEN?

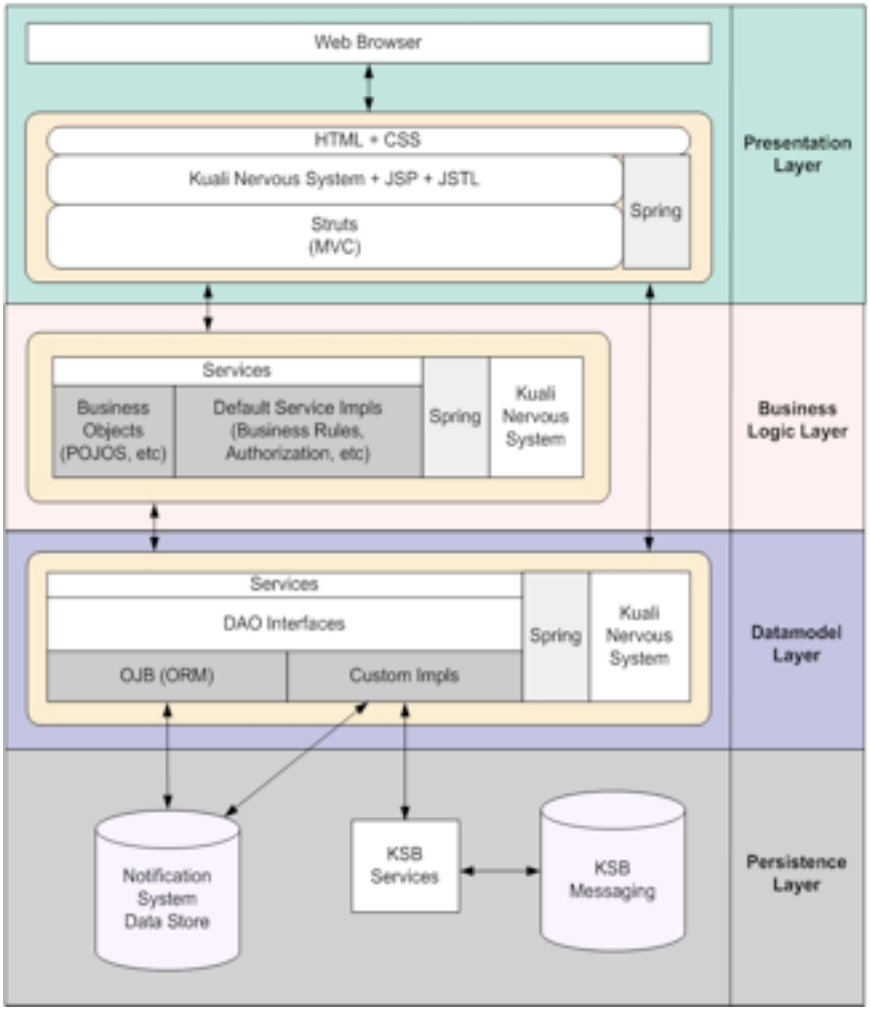
Kuali Enterprise Notification is a form of communication between distributed systems that allows messages to be sent securely and consistently. These messages act as notifications upon receipt and are processed asynchronously within the service layer. The following architectural diagram represents the flow of messages in a typical Rice Environment.

Figure 1.1. KEN Message Flow



From a developer's perspective the diagram below helps to represent the inner workings of how KEN stores data from the Data Modeling Layer into the Persistence Layer.

Figure 1.2. KEN Message Storage



The following sections of documentation aim at describing the inner workings of KEN as well as how those pieces interact with Rice, specifically KEW. KEN itself is an interface that sits on top of KEW's API. This allows for registration and publishing of notifications, which then flow through KEW to result in a KEW action request. See KEW Overview for more information. In addition to the action list, KEW can be optionally configured to forward these requests to the Kuali Communications Broker or KCB for short. This module is logically related to KEN and handles dispatching messages based on the user preferences. Once messages are dispatched, a response or acknowledgement can be created.

Chapter 2. KEN Configuration Parameters

Table 2.1. KEN Core Parameters

Configuration Parameter	Description	Default value
ken.url	The base URL of the KEN webapp; this should be changed when deploying for external access	${application.url}/ken
notification.resolveMessageDeliveriesJob.startDelayMS	The start delay (in ms) of the job that resolves message deliveries	5000
notification.resolveMessageDeliveriesJob.intervalMS	The interval (in ms) between runs of the message delivery resolution job	10000
notification.processAutoRemovalJob.startDelayMS	The start delay (in ms) of the job that auto-removes messages	60000
notification.processAutoRemovalJob.intervalMS	The interval (in ms) between runs of the message auto-removal job	60000
notification.quartz.autostartup	Whether to automatically start the KEN Quartz jobs	true
notification.concurrent.jobs	Whether the invocation of a KEN Quartz job can overlap another KEN Quartz job running concurrently	true
ken.system.user	The principal name of the user that KEN should use when initiating KEN-originated documents	notsys
kcb.url	The base URL of the KCB (notification broker) webapp	${application.url}/kcb
kcb.messaging.synchronous	Whether notification messages are processed synchronously	false
kcb.messageprocessing.startDelayMS	The start delay (in ms) of the job that processes notification messages	50000
kcb.messageprocessing.repeatIntervalMS	The interval (in ms) between runs of the notification message processing job	30000
kcb.quartz.group	Group name of the KCB Quartz job	KCB-Delivery
kcb.quartz.job.name	Name of the KCB Quartz job	MessageProcessingJobDetail
kcb.maxProcessAttempts	Maximum number of times that KCB will attempt to process a notification message	3
notification.processUndeliveredJob.intervalMS	The elapsed time, in milliseconds, between runs of the KEN process undelivered notifications job.	10000
notification.processUndeliveredJob.startDelayMS	The elapsed time, in milliseconds, between the start of the application and the first run of the KEN process undelivered notifications job.	10000

Note

As of Rice 1.0.1, The parameter **kcb.smtp.host** is no longer used. The smtp server settings that are required for sending email notifications with KEN are documented in the Configuration section of the Kuali Enterprise Workflow Guide under the subsection **Email Configuration.**

Chapter 3. KEN Channels

A KEN Channel is correlated to a specific type of notification. An example of a Channel's use may be to send out information about upcoming Library Events or broadcast general announcements on upcoming concerts. Channels are subscribed to in the act of receiving notifications from a publisher or producer. They can also be unsubscribed to and removed from the data store from within the UI. The Channel Definitions are stored in the database table KREN_CHNL_T. The columns are listed as follows:

Table 3.1. KREN_CHNL_T

Column	Description
CHNL_ID	Identifier for the Channel
NM	Name of the Channel represented in the UI
DESC_TXT	Description of the Channel
SUBSCRB_IND	Determines if the Channel can or cannot be subscribed to from the UI. This also determines if the channel will be displayed in the UI
VER_NBR	Version Number for the Channel

Channel Subscription

Subscribing to a notification channel causes all notifications sent on the channel to be delivered to the subscriber, regardless of the list of users or groups specified as recipients of the notification.

Subscriptions are divided into two categories:

- **Subscribers** – Users can subscribe to any channel where the Subscription Flag is set to "Yes".
- **Default Recipients** – Each channel has a list of users and groups that receive all channel notifications. To view the complete list of default recipients for a channel, run the following query replacing ? with your channel id:

```
SELECT * FROM KREN_RECIP_LIST_T
WHERE CHNL_ID = '?';
```

Channels can be subscribed to through the UI and also through the direct access to the data store. To add a channel that can be subscribed to simply run the following SQL statement against the data store customizing value entries to your needs:

```
INSERT INTO KREN_CHNL_T (CHNL_ID,DESC_TXT,NM,SUBSCRB_IND,VER_NBR)
VALUES (2, 'This channel is used for sending out information about Library Events.', 'Library Events Channel',
 'Y', 1)
```

To add default recipients to a channel, first locate the recipients through the Person Lookup function as described here in the KIM Guide.

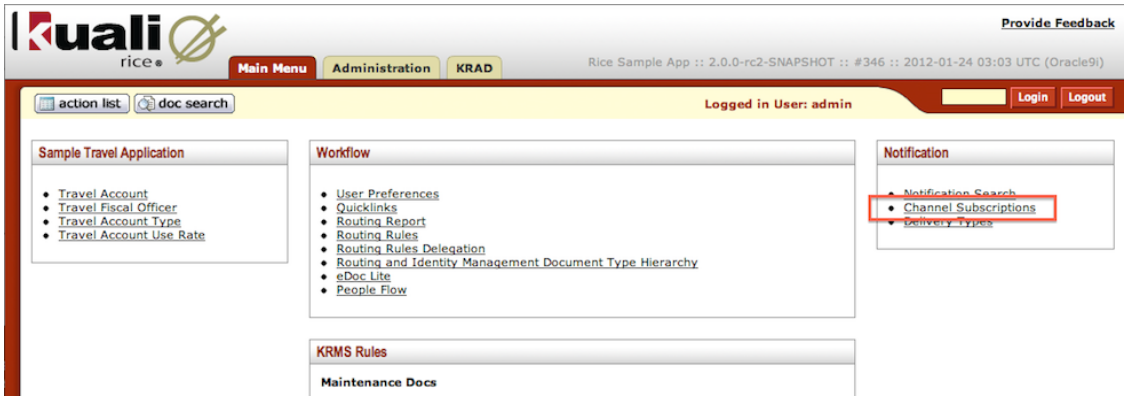
Then, for each person's Principal ID, execute the following SQL where 'X' is the next RECIP_LIST_ID, 'Y' is the channel ID, and 'Z' is the Principal ID:

```
INSERT INTO KREN_RECIP_LIST_T (RECIP_LIST_ID, CHNL_ID, RECIP_TYP_CD, RECIP_ID, OBJ_ID, VER_NBR)
VALUES ('X', 'Y', 'USER', 'Z', '', 1);
```

Managing Subscribers

Users can manage their subscriptions to channels where the Subscription Flag is **Yes** using the Channel Subscriptions form. On the Main Menu, click the **Channel Subscriptions** link in the **Notification** box:

Figure 3.1. Notification Channel: Channel Subscriptions



This displays the list of channels that allow the user to manage their subscriptions:

Figure 3.2. Chanel Subscriptions: Manage

Click the **Subscribe** button for a channel to subscribe to a channel for which you are not already subscribed. Click the **Unsubscribe** button to cancel your subscription to the channel.

Notification Channel

A notification channel is a communication stream used as a means to organize notifications by topic or audience. Users and groups can subscribe to a channel to receive the notifications that interest them.

You can think of a notification channel as being like a television channel. Producers put messages on a channel, like producers at television stations air programs. Subscribers receive the messages, like television

viewers watch the programs. Unlike a television program, a message sent on a channel will wait for you to read it; you do not need to be actively reading the channel to avoid missing a message.

Each channel has the following attributes:

- **ID** – This numeric value is used in various tables to identify the channel. Each channel must have a unique ID, but there is no requirement for the IDs to be consecutive or to start with a particular value.
- **Name** – This is the text displayed to the user in the user interface. Each channel must have a unique name.
- **Description** – This is text which further describes the channel.
- **Subscription Flag** – This flag identifies whether users can adjust their subscription status on the channel. If "Yes", users can see the channel in the "Channel Subscriptions" form and change their subscription status. If "No", a user's subscription to the channel is managed by other means (for example, group membership).
- **Producers** – This is a list of users who are authorized to send notifications on the channel.
- **Default Recipient List** – This is a list of users and groups who will receive notifications sent to the channel, regardless of channel subscriptions or notification-specific recipients.
- **Reviewers** – This is a list of users who are responsible for approving notifications sent to the channel.

Chapter 4. KEN Producers

A KEN Producer submits notifications for processing through the system. An example of a Producer would be a mailing daemon that represents messages sent from a University Library System.

Characteristics of a Producer:

- Producers create and send notifications to a specific destination through various Channels.
- Each Producer contains a list of Channels that it may send notifications to.
- Producer Definitions are stored in the database table KREN_PRODCR_T.

Table 4.1. KREN_PRODCR_T

Column	Description
CNTCT_INFO	The email address identifying the Producer of the Notification.
DESC_TXT	A Description of the Producer.
NM	Name of the Producer.
PRODCR_ID	The Producer's Channel Identifier. See the KREN_CHNL_PRODCR_T table found in the database for more information on how Producers link to Channels.
VER_NBR	Version Number for the Producer.

Adding Producers

The Producer can be added through direct access to the data store. To add a Producer run the following SQL statement against the data store customizing value entries to your needs:

```
INSERT INTO KREN_PRODCR_T (CNTCT_INFO,DESC_TXT,NM,PRODCR_ID,VER_NBR)
  VALUES ('kuali-ken-testing@cornell.edu','This producer represents messages sent from the general message
 sending forms.','Notification System',1,1)
```

Chapter 5. KEN Content Types

Overview

A Content Type is part of the message content of a notification that may be sent using KEN. It can be as simple as a single message string, or something more complex, such as an event that might have a date associated with it, start and stop times, and other metadata you may want to associate with the notification.

KEN is distributed with two Content Types: Simple and Event.

Warning

It is strongly recommended that you leave these two Content Types intact, but you can use them as templates for creating new Content Types.

Every notification sent through KEN must be associated with a **registered** Content Type. Registration of Content Types requires administrative access to the system and is described in the KEN Content Types section in the User Guide. The rest of this section describes the Content Type attributes that are required for registration.

Content Type Attributes

A Content Type is represented as a *NotificationContent* business object and consists of several attributes, described below:

id - Unique identifier that KEN automatically creates when you add a Content Type

name - This is a unique string that identifies the content. For example, *ItemOverdue* might be the *name* used for a notification Content Type about an item checked out from the campus library.

description - This is a more verbose description of the Content Type. For example, "Library item overdue notices" might be the *description* for *ItemOverdue*.

namespace - This is the string used in the XSD schema and XML to provide validation of the content, for example, *notification/ContentTypeItemOverdue*. The XSD namespace is typically the *name* attribute concatenated to the *notification/ContentType* string. Note how it is used in the **XSD** and **XSL** examples below.

xsd - The XSD attribute contains the complete W3C XML Schema compliant code.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This schema defines a generic event notification type in order for it to be accepted into the system. -->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:c="ns:notification/common"
        xmlns:ce="ns:notification/ContentTypeItemOverdue"
        targetNamespace="ns:notification/ContentTypeItemOverdue"
        attributeFormDefault="unqualified"
        elementFormDefault="qualified">
    <annotation>
        <documentation xml:lang="en">Item Overdue Schema</documentation>
    </annotation>
    <import namespace="ns:notification/common" schemaLocation="resource:notification/notification-common" />

    <!-- The content element describes the content of the notification.  It contains a message (a simple
 String) and a message element -->
    <element name="content">
        <complexType>
```

```
            <sequence>
                <element name="message" type="c:LongStringType"/>
                <element ref="ce:event"/>
            </sequence>
        </complexType>
    </element>

    <!-- This is the itemoverdue element.  It describes an item overdue notice containing a summary,
 description, location, due date, and the amount of the fine levied -->
    <element name="itemoverdue">
        <complexType>
            <sequence>
                <element name="summary" type="c:NonEmptyShortStringType" />
                <element name="description" type="c:NonEmptyShortStringType" />
                <element name="location" type="c:NonEmptyShortStringType" />
                <element name="dueDate" type="dateTime" />
                <element name="fine" type="decimal" />
            </sequence>
        </complexType>
    </element>
</schema>
```

xsl - The XSD attribute contains the complete XSL code that will be used to transform a notification in XML to html for rendering in an Action List.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- style sheet declaration: be very careful editing the following, the
 default namespace must be used otherwise elements will not match -->
<xsl:stylesheet

    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:n="ns:notification/ContentTypeEvent"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="ns:notification/ContentTypeItemOverdue resource:notification/ContentTypeItemOverdue"
    exclude-result-prefixes="n xsi">



    <!-- output an html fragment -->
    <xsl:output method="html" indent="yes" />


    <!-- match everything -->
    <xsl:template match="/n:content" >
        <table class="bord-all">
            <xsl:apply-templates />
        </table>
    </xsl:template>


    <!--  match message element in the default namespace and render as strong -->
    <xsl:template match="n:message" >
        <caption>
            <strong><xsl:value-of select="." disable-output-escaping="yes"/></strong>
        </caption>
    </xsl:template>


    <!-- match on itemoverdue in the default namespace and display all children -->
    <xsl:template match="n:itemoverdue">
        <tr>
            <td class="thnormal"><strong>Summary: </strong></td>
            <td class="thnormal"><xsl:value-of select="n:summary" /></td>
        </tr>
        <tr>
            <td class="thnormal"><strong>Item Description: </strong></td>
            <td class="thnormal"><xsl:value-of select="n:description" /></td>
        </tr>
        <tr>
            <td class="thnormal"><strong>Library: </strong></td>
            <td class="thnormal"><xsl:value-of select="n:location" /></td>
        </tr>
        <tr>
```

```
            <td class="thnormal"><strong>Due Date: </strong></td>
            <td class="thnormal"><xsl:value-of select="n:startDateTime" /></td>
        </tr>
        <tr>
            <td class="thnormal"><strong>Fine: </strong></td>
            <td class="thnormal">$<xsl:value-of select="n:fine" /></td>
        </tr>
    </xsl:template>
</xsl:stylesheet>
```

Chapter 6. KEN Notifications

This document provides information about the attributes of a Notification. These attributes are elements such as message content, who is sending the notification, who should receive it, etc. Kuali Enterprise Notification (KEN) supports an arbitrary number of Content Types, such as a simple message or an event notification. Each Content Type consists of a common set of attributes and a content attribute.

Common Notification Attributes

Table 6.1. Common Notification Attributes

Name	Type	Required	Description	Example
channel	string	yes	• Name of a channel • Must be registered	Library Events
producer	string	yes	• Name of the producing system • Must be registered and given authority to send messages on behalf of the <*Library Events*> channel	Library Calendar System
senders	a list of strings	yes	A list of the names of people on whose behalf the message is being sent	TestUser1, TestUser2
recipients	a list of strings	yes	A list of the names of groups or users to whom the message is being sent	library-staff-group, TestUser1, TestUser2
deliveryType	string	yes	fyi or ack	fyi
sendDateTime	datetime	no	When to send the notification	2006-01-01 00:00:00.0
autoRemoveDateTime	datetime	no	When to remove the notification	2006-01-02 00:00:00.0
priority	string	yes	An arbitrary priority; these must be registered in KEN; the system comes with defaults of *normal*, *low*, and *high*	normal
contentType	string	yes	Name for the content; KEN comes set up with *simple* and *event*; new contentTypes must be registered in KEN	simple
content	see below	yes	The actual content	see below
docTypeName	string	no	The name of a custom KEW document type to be used to route this Notification with different security parameters. Typically different than the default "KualiNotification" and can be left blank if the default is desired.	KualiNotification

Notification Priority

The priority of a notification indicates its importance. It has no effect on how KEN processes the notification, but KEN can use it when delivering a message to determine how to present the notification.

Each priority has these attributes:

- **ID** – This numeric value defines the order that KEN displays the priorities in the user interface. The lowest ID is displayed at the top of the selection field and is the default value. The remaining priorities are listed in the selection field in ascending ID order. Each priority must have a unique ID, but there is no requirement for the IDs to be consecutive or to start with a particular value.
- **Name** – This is the text displayed to the user in the user interface. Each priority must have a unique name.
- **Description** – This text further describes the priority.

- **Order** – This numeric value determines the relative importance of the priority, with lower order numbers indicating a higher importance. Although not required, each priority should have a unique order value. There is no requirement for the order values to be consecutive or to start with a particular value.
- **Version** – This numeric value lets you perform optimistic locking on the database row. It is initialized to one when the row is created and should be incremented each time the row is updated.

By default, three priorities are defined in KEN:

Table 6.2. Notification: Priority Attributes

ID	Name	Description	Order
1	Normal	Normal priority	2
2	Low	A low priority	3
3	High	A high priority	1

Message Content

Notifications are differentiated using the *contentType* attribute and the contents of the *content* element. The *content* element can be as simple as a message string or it may be a complex structure. For example, a simple notification may only contain a message string, whereas an *Event* Content Type might contain a summary, description, location, and start and end dates and times. Examples of the *Simple* and *Event* Content Types:

Sample XML for a Simple Notification

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- A Simple Notification Message -->
<notification xmlns="ns:notification/NotificationRequest"
    xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
    xsi:schemaLocation="ns:notification/NotificationRequest
    resource:notification/NotificationRequest">
    <!-- this is the name of the notification channel -->
    <!-- that has been registered in the system -->
    <channel>Campus Status Announcements</channel>


    <!-- this is the name of the producing system -->
    <!-- the value must match a registered producer -->
    <producer>Campus Announcements System</producer>

    <!-- these are the people that the message is sent on -->
    <!-- behalf of -->
    <senders>
        <sender>John Fereira</sender>
    </senders>


    <!-- who is the notification going to? -->
    <recipients>
        <group>Everyone</group>
        <user>jaf30</user>
    </recipients>


    <!--  fyi or acknowledge -->
    <deliveryType>fyi</deliveryType>


    <!-- optional date and time that a notification should be sent -->
    <!-- use this for scheduling a single future notification to happen -->
    <sendDateTime>2006-01-01T00:00:00</sendDateTime>
```

```
    <!-- optional date and time that a notification should be removed -->
    <!-- from all recipients' lists, b/c the message no longer applies -->
    <autoRemoveDateTime>3000-01-01T00:00:00</autoRemoveDateTime>


    <!-- this is the name of the priority of the message -->
    <!-- priorities are registered in the system, so your value -->
    <!-- here must match one of the registered priorities -->
    <priority>Normal</priority>


    <title>School is Closed</title>


    <!-- this is the name of the content type for the message -->
    <!-- content types are registered in the system, so your value -->
    <!-- here must match one of the registered contents -->
    <contentType>Simple</contentType>


    <!-- actual content of the message -->
    <content xmlns="ns:notification/ContentTypeSimple"
        xsi:schemaLocation="ns:notification/ContentTypeSimple
        resource:notification/ContentTypeSimple">

        <message>Snow Day! School is closed.</message>
    </content>
</notification>
```

Sample XML for an Event Notification

```
<?xml version="1.0" encoding="UTF-8"?>


<notification xmlns="ns:notification/NotificationMessage"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="ns:notification/NotificationMessage
    resource:notification/NotificationMessage">
    <!-- this is the name of the notification channel -->
    <!-- that has been registered in the system -->
    <channel>Concerts Coming to Campus</channel>


    <!-- this is the name of the producing system -->
    <!-- the value must match a registered producer -->
    <producer>Campus Events Office</producer>


    <!-- these are the people that the message is sent on -->
    <!-- behalf of -->
    <senders>
        <sender>ag266</sender>
        <sender>jaf30</sender>
    </senders>


    <!-- who is the notification going to? -->
        <recipients>
            <group>Group X</group>
            <group>Group Z</group>
            <user>ag266</user>
            <user>jaf30</user>
            <user>arh14</user>
        </recipients>


    <!--  fyi or acknowledge -->
    <deliveryType>fyi</deliveryType>


    <!-- optional date and time that a notification should be sent -->
    <!-- use this for scheduling a single future notification to happen -->
    <sendDateTime>2006-01-01 00:00:00.0</sendDateTime>
```

```
    <!-- optional date and time that a notification should be removed -->
    <!-- from all recipients' lists, b/c the message no longer applies -->
    <autoRemoveDateTime>2007-01-01 00:00:00.0</autoRemoveDateTime>


    <!-- this is the name of the priority of the message -->
    <!-- priorities are registered in the system, so your value -->
    <!-- here must match one of the registered priorities -->
    <priority>Normal</priority>


    <!-- this is the name of the content type for the message -->
    <!-- content types are registered in the system, so your value -->
    <!-- here must match one of the registered contents -->
    <contentType>Event</contentType>


    <!-- actual content of the message -->
    <content>
        <message>CCC presents The Strokes at Cornell</message>


        <!-- an event that it happening on campus -->
        <event xmlns="ns:notification/ContentEvent"
            xsi:schemaLocation="ns:notification/ContentEvent
            resource:notification/ContentEvent">
            <summary>CCC presents The Strokes at Cornell</summary>
            <description>blah blah blah</description>
            <location>Barton Hall</location>
            <startDateTime>2006-01-01T00:00:00</startDateTime>
            <stopDateTime>2007-01-01T00:00:00</stopDateTime>
        </event>
    </content>
</notification>
```

Notification Response

When KEN sends a notification, it always returns a response. This is an outline in XML of that response:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
    <status>success</status>
</response>
```

Chapter 7. Enterprise Notification Priority

Managing Priorities

There is no user interface page to manage priorities so you must make changes to the list of priorities in the **kren_prio_t** table using SQL.

The table has these columns:

Table 7.1. KREN_PRIO_T

Name	Type	Max Size	Required	Attribute
PRIO_ID	Numeric	8	Yes	ID
NM	Text	40	Yes	Name
DESC_TXT	Text	500	Yes	Description
PRIO_ORD	Numeric	4	Yes	Order
VER_NBR	Numeric	8	Yes	Version

Example 7.1. Example – This is an example of how to add a Priority into the table:

```
INSERT INTO kren_prio_t (PRIO_ID, NM, DESC_TXT, PRIO_ORD, VER_NBR) VALUES (8, 'Bulk', 'Mass notifications', 6,
 1);
```

Chapter 8. KEN Delivery Types

This section describes Kuali Enterprise Notification (KEN) Delivery Types, or what are sometimes called Message Deliverers. A Message Deliverer Plugin is the mechanism used to deliver a notification to end users. All notifications sent through KEN appear in the Action List for each recipient for which the notification is intended. This message also contains an Email Delivery Type that allows you to send end users a notification summary as an email message. Note that for a Delivery Type other than the default (KEWActionList), the content of the notification is typically just a summary of the full notification.

Implementing the Java Interface

Creating a new Delivery Type primarily involves implementing a Java interface called **org.kuali.rice.kew.deliverer.NotificationMessageDeliverer**. The source code of the interface:

```
/*
 * Copyright 2007 The Kuali Foundation
 *
 * Licensed under the Educational Community License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.opensource.org/licenses/ecl2.php
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.

 */
package org.kuali.rice.ken.deliverer;



import java.util.HashMap;

import java.util.LinkedHashMap;


import org.kuali.rice.ken.bo.NotificationMessageDelivery;
import org.kuali.rice.ken.exception.ErrorList;

import org.kuali.rice.ken.exception.NotificationAutoRemoveException;
import org.kuali.rice.ken.exception.NotificationMessageDeliveryException;
import org.kuali.rice.ken.exception.NotificationMessageDismissalException;


/**
 * This class represents the different types of Notification Delivery Types that the system can handle.
 * For example, an instance of delivery type could be "ActionList" or "Email" or "SMS".  Any deliverer
 implementation
 * adhering to this interface can be plugged into the system and will be automatically available for use.
 * @author Kuali Rice Team (kuali-rice@googlegroups.com)
*/

public interface NotificationMessageDeliverer {
  /**
   * This method is responsible for delivering the passed in messageDelivery record.
   * @param messageDelivery The messageDelivery to process
   * @throws NotificationMessageDeliveryException
  */

  public void deliverMessage(NotificationMessageDelivery messageDelivery) throws
 NotificationMessageDeliveryException;
    /**
     * This method handles auto removing a message delivery from a person's list of notifications.
     * @param messageDelivery The messageDelivery to auto remove
```

```
    * @throws NotificationAutoRemoveException
    */

 public void autoRemoveMessageDelivery(NotificationMessageDelivery messageDelivery) throws
NotificationAutoRemoveException;
   /**
    * This method dismisses/removes the NotificationMessageDelivery so that it is no longer being presented to
the user
    * via this deliverer.  Note, whether this action is meaningful is dependent on the deliverer
implementation.  If the
    * deliverer cannot control the presentation of the message, then this method need not do anything.
    * @param messageDelivery the messageDelivery to dismiss
    * @param the user that caused the dismissal; in the case of end-user actions, this will most likely be the
user to
    *   which the message was delivered (user recipient in the NotificationMessageDelivery object)
    * @param cause the reason the message was dismissed
    */


 public void dismissMessageDelivery(NotificationMessageDelivery messageDelivery, String user, String cause)
throws NotificationMessageDismissalException;
```

Default Delivery Types

To find and configure the default delivery types configured for Rice, on the Main Menu tab, under the Notification area, click on Delivery Types.

Figure 8.1. Find Delivery Types

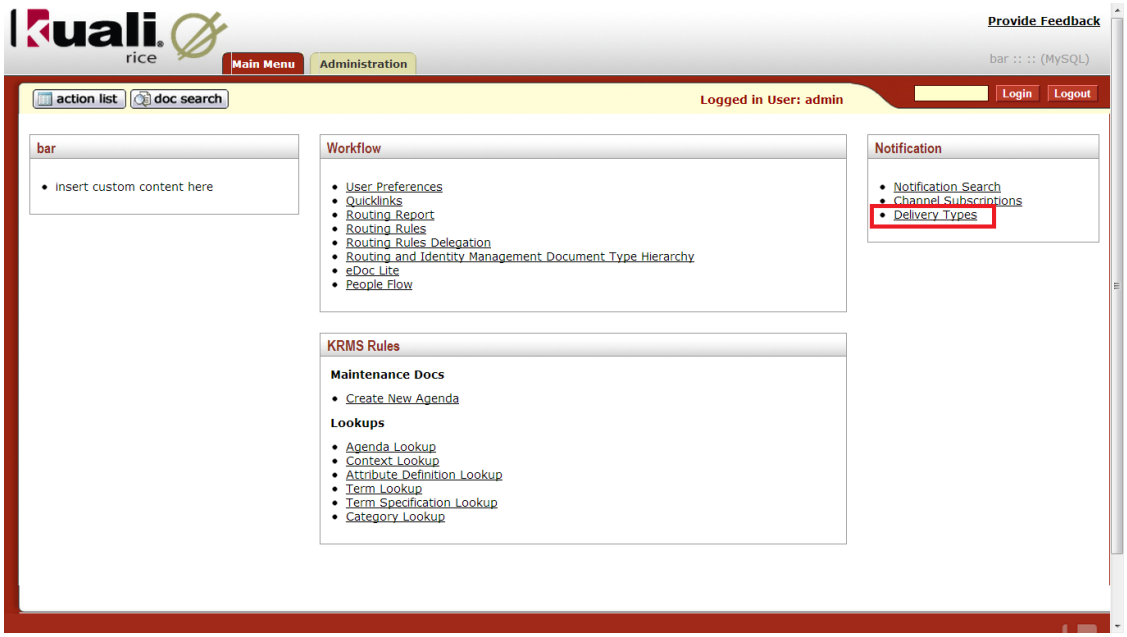
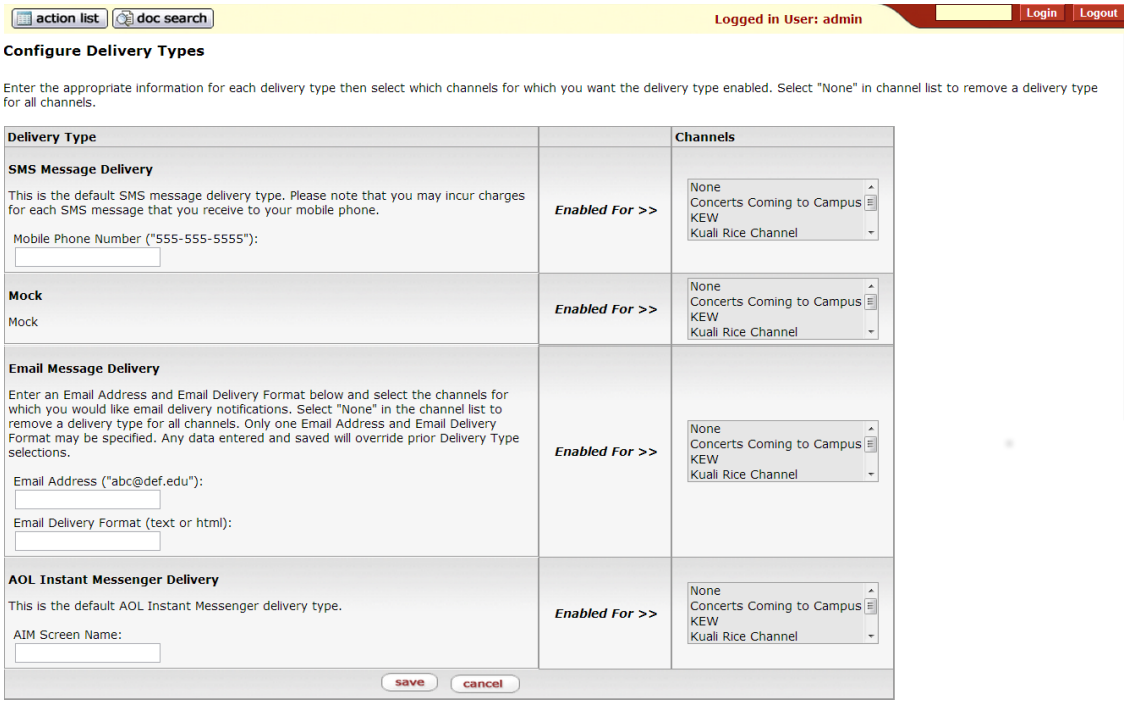


Figure 8.2. List and Configure Delivery Types



Chapter 9. KEN: Sending a Notification

The Kuali Enterprise Notification system (KEN) provides for a way to programmatically send a notification. An application may construct a notification using the KEN web service API.

Alternatively, users can use the user interface (UI) to send notifications.

Warning

When you send a notification, KEN generates a document for every person you send it to. So if you send it to a work group or a role with n members, you'll instantaneously create n number of workflow documents. Some workgroups or roles may have a huge number of members so this feature needs to be taken into consideration.

Send a Notification Using the Web Service API

To send a notification using the web service API, the notification must be constructed as an XML document that validates against a schema for a specific Content Type. For more detail, see the Notifications documentation.

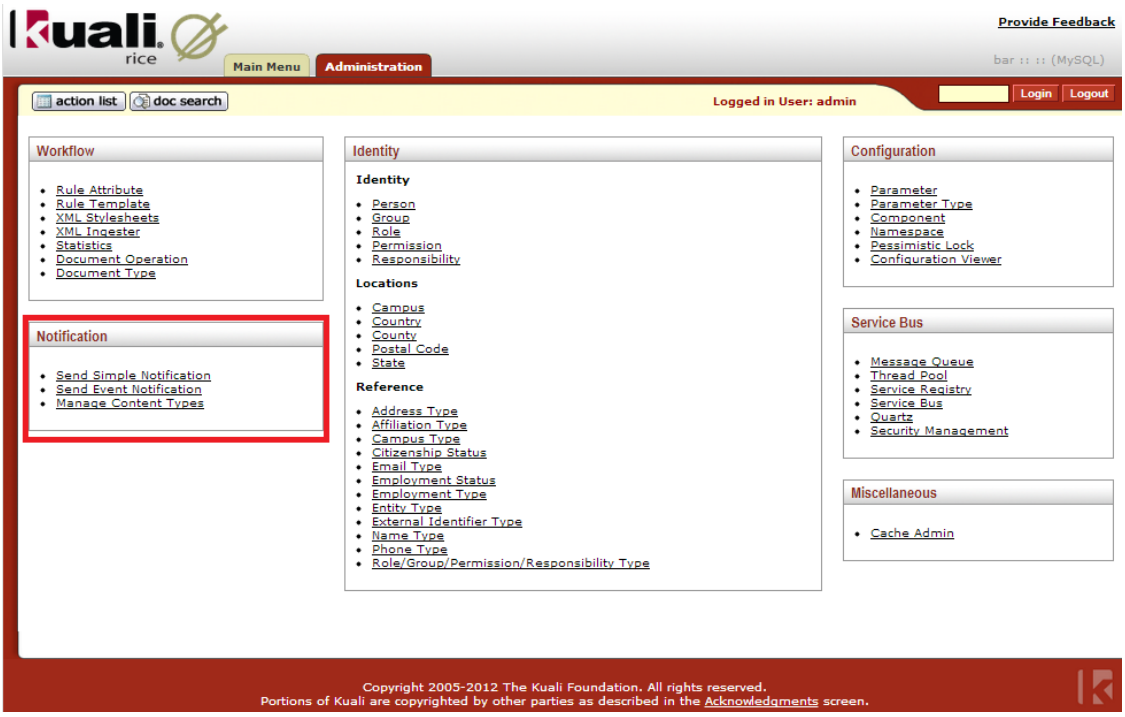
To validate your notification XML, you must construct the XSD schema filename. To construct this file name, append the Content Type value to *ContentType*.

For example, if you create a new Content Type for a library book overdue notification, then the *contentType* element value should be *OverdueNotice* and the schema file you created for validation of the notification XML should be **ContentTypeOverdueNotice.xsd**. This XML schema should be declared as a namespace in the **content** element of the notification XML. Out of the box, KEN comes with *Simple* and *Event* Content Types.

Send a Notification Using the UI

To send a notification via the user interface (UI), start at the Administration tab.

Figure 9.1. Notification Window



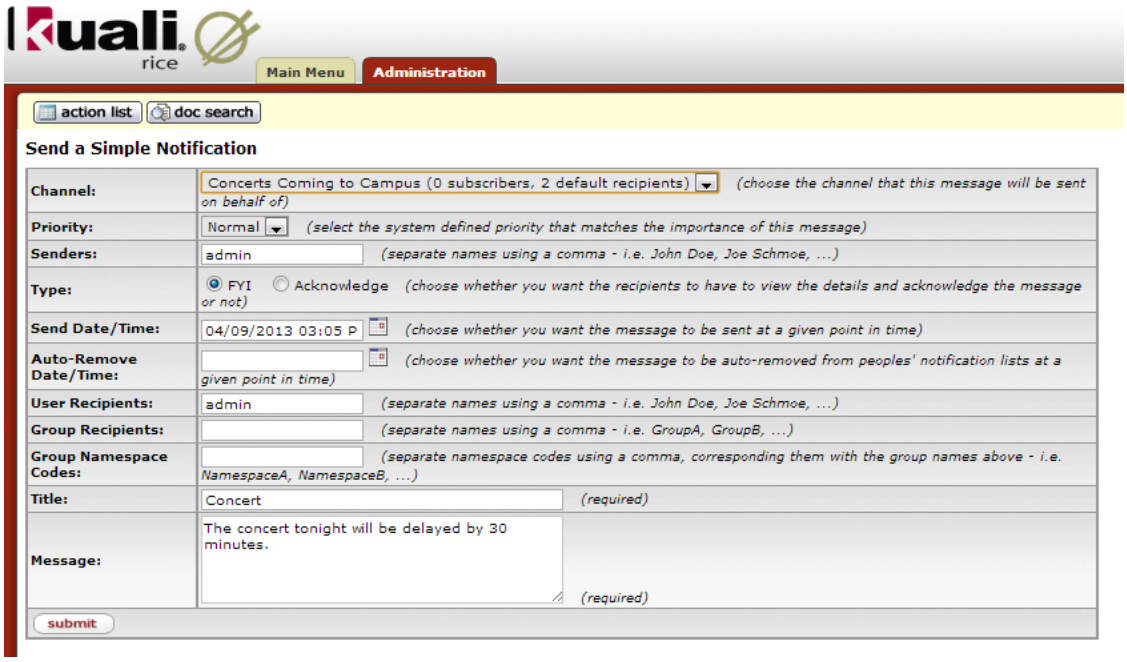
In the Notification section there are 3 options:

- Send Simple Notification
- Send Event Notification
- Manage Content Types

Each of these options will be examined in the following sub sections.

Send Simple Notification

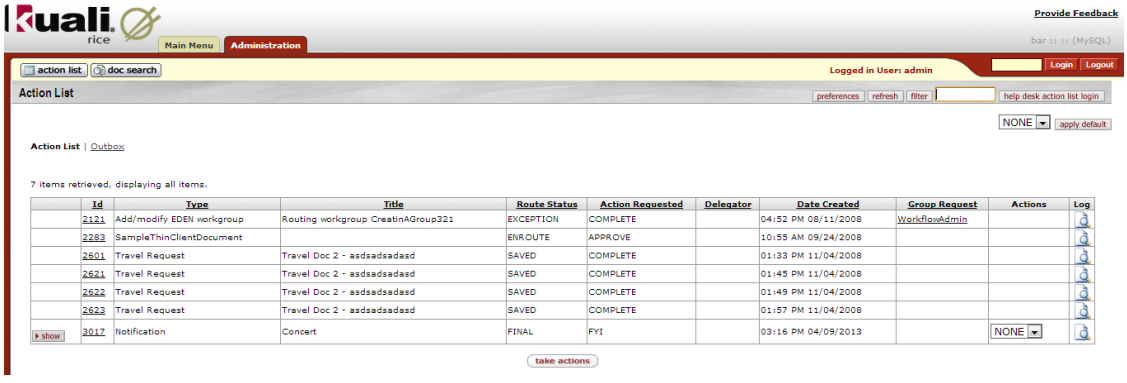
Figure 9.2. Send Simple Notification



Upon hitting **`submit`**, any errors or a message "Notification(s) sent" will appear just above the dialog box.

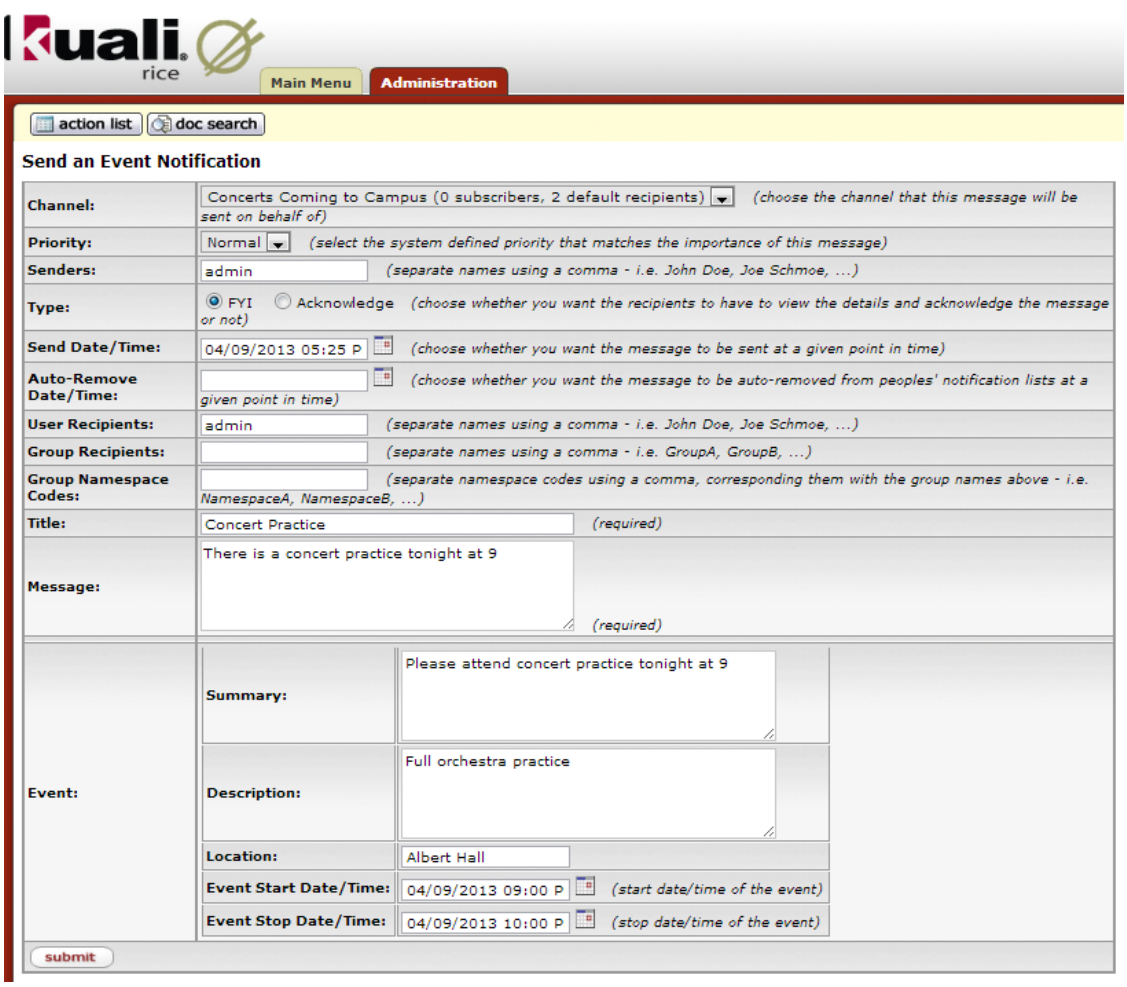
To see the notification message that was sent, in this example, to the user "admin", after logging into the system as admin, click on the **`action list`** button. Since the list is sorted in ascending date created order, you will find the message at the bottom of the list.

Figure 9.3. Action List 1



Send Event Notification

Figure 9.4. Send Event Notification



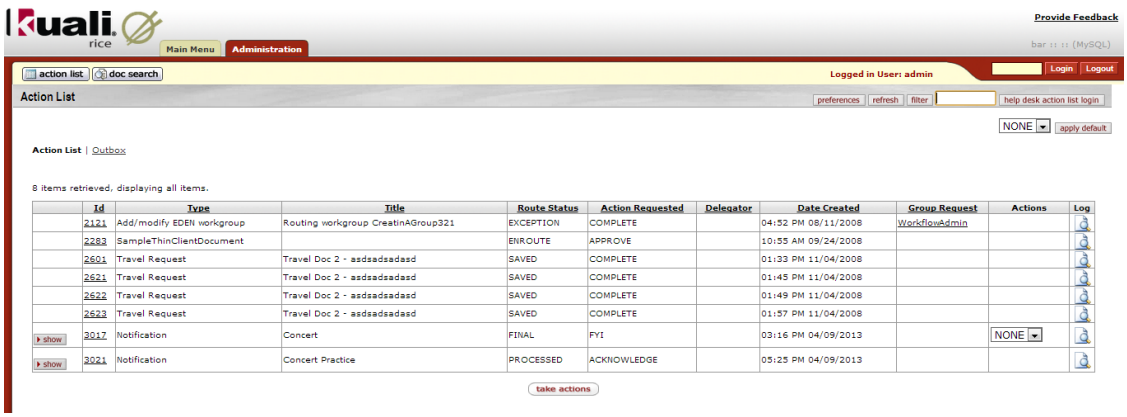
Upon hitting **`submit`**, any errors or a message "Notification(s) sent" will appear just above the dialog box.

To see the notification message that was sent, in this example, to the user "admin", after logging into the system as admin, click on the **`action list`** button. Since the list is sorted in ascending date created order, you will find the message at the bottom of the list.

Note

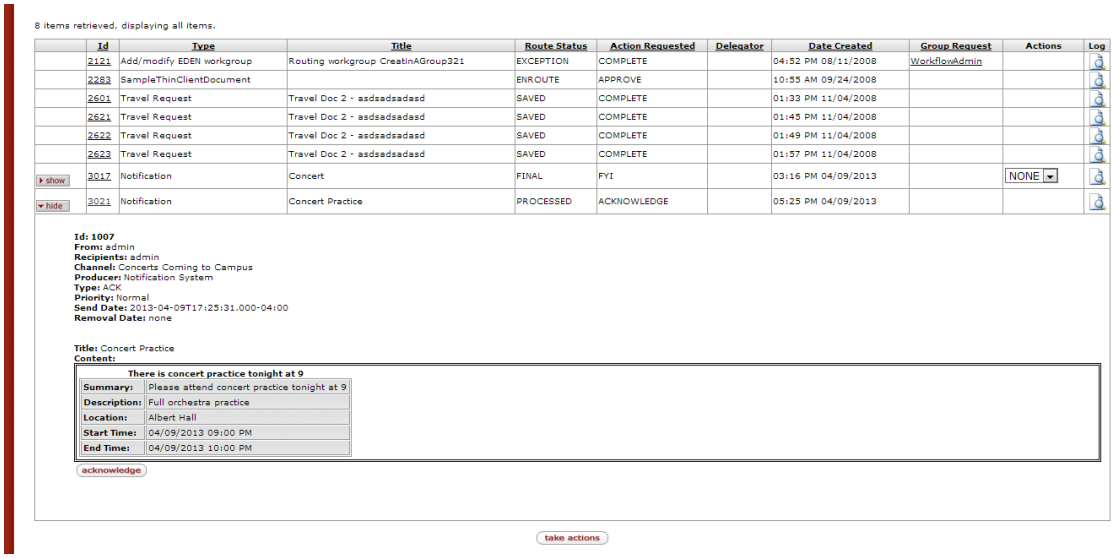
The action requested in this case is **acknowledge** instead of FYI.

Figure 9.5. Action List 2



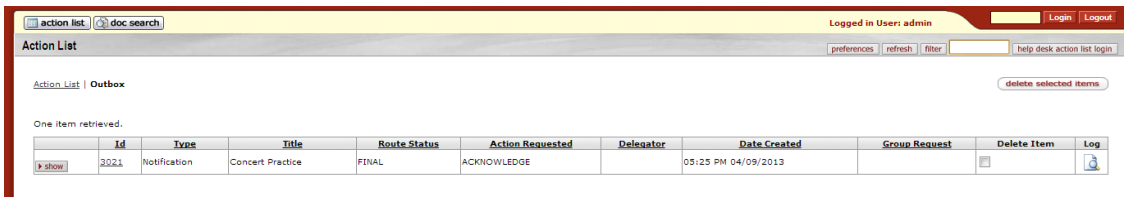
Clicking on **Show** will display additional information about the event as in the figure below.

Figure 9.6. Show Event Detail



Clicking the **acknowledge** button removes the event item from the action list, and places it in the Outbox for the user. Alternatively, clicking the **take actions** button will take all the outstanding appropriate action for each item in the Action List. Clicking the **Outbox** link will display the item there, as shown below. From here, the user can select the **Delete Item** checkbox and click the **delete selected items** button to remove it.

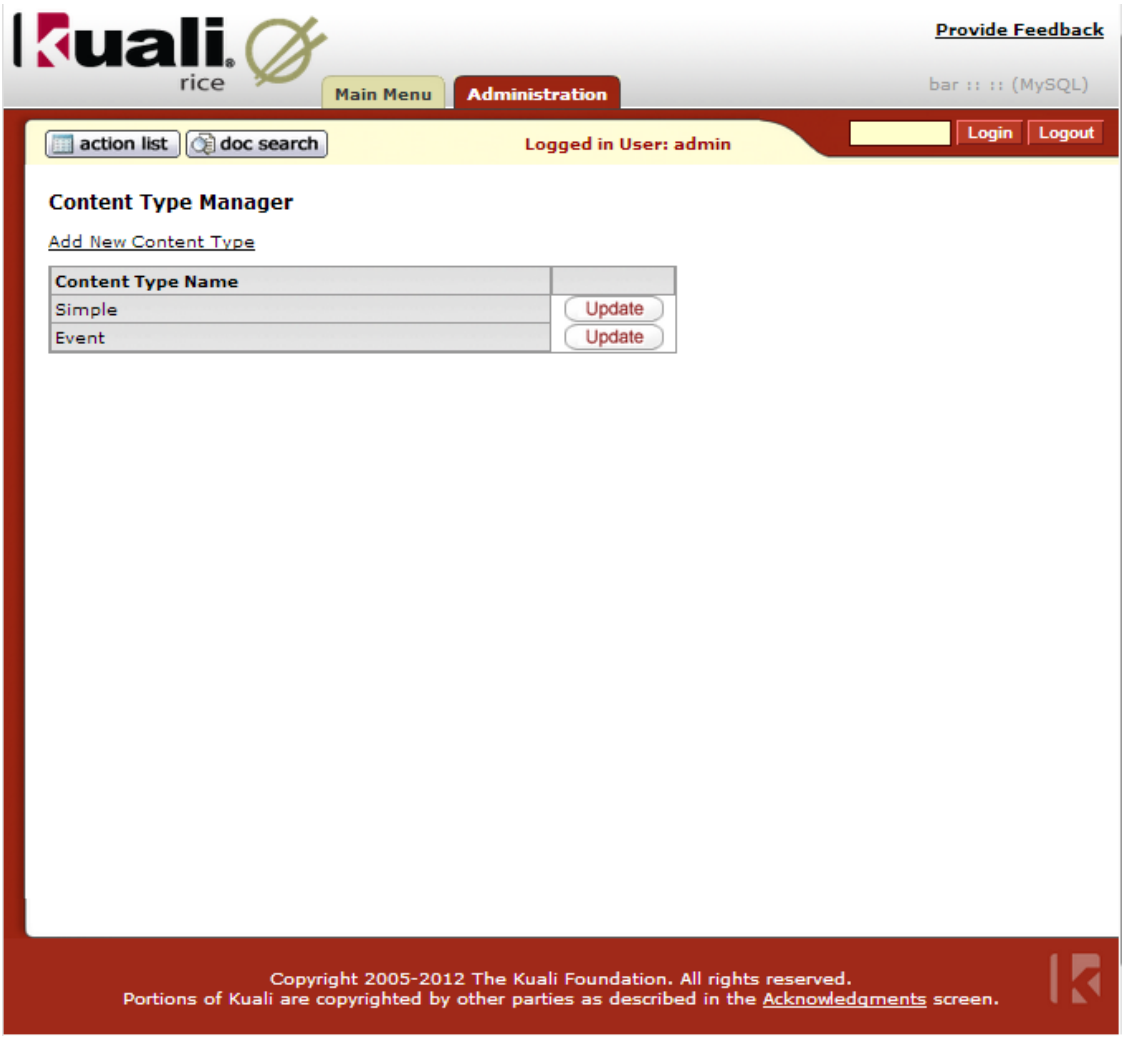
Figure 9.7. Outbox



Manage Content Types

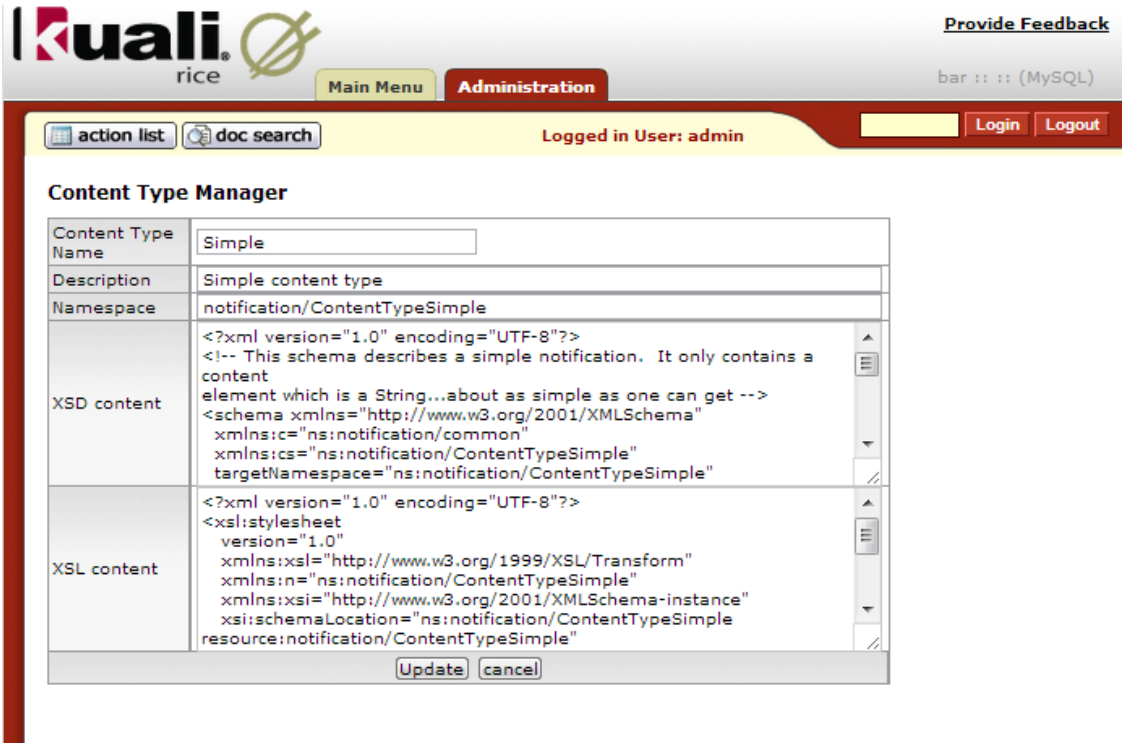
Clicking on the Manage Content Types link brings up the following screen.

Figure 9.8. Content Type Manager



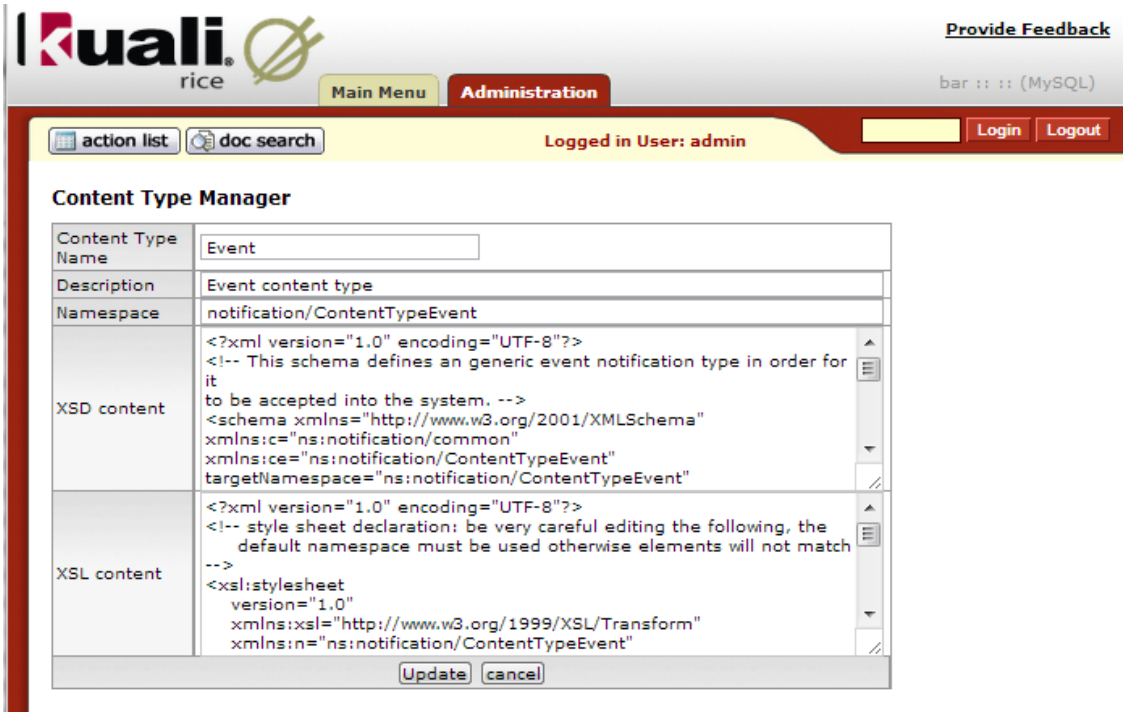
Clicking on the **`Update`** next to the **`Simple`** entry brings up the following screen where XSD and XSL content may be updated.

Figure 9.9. Simple Content Type



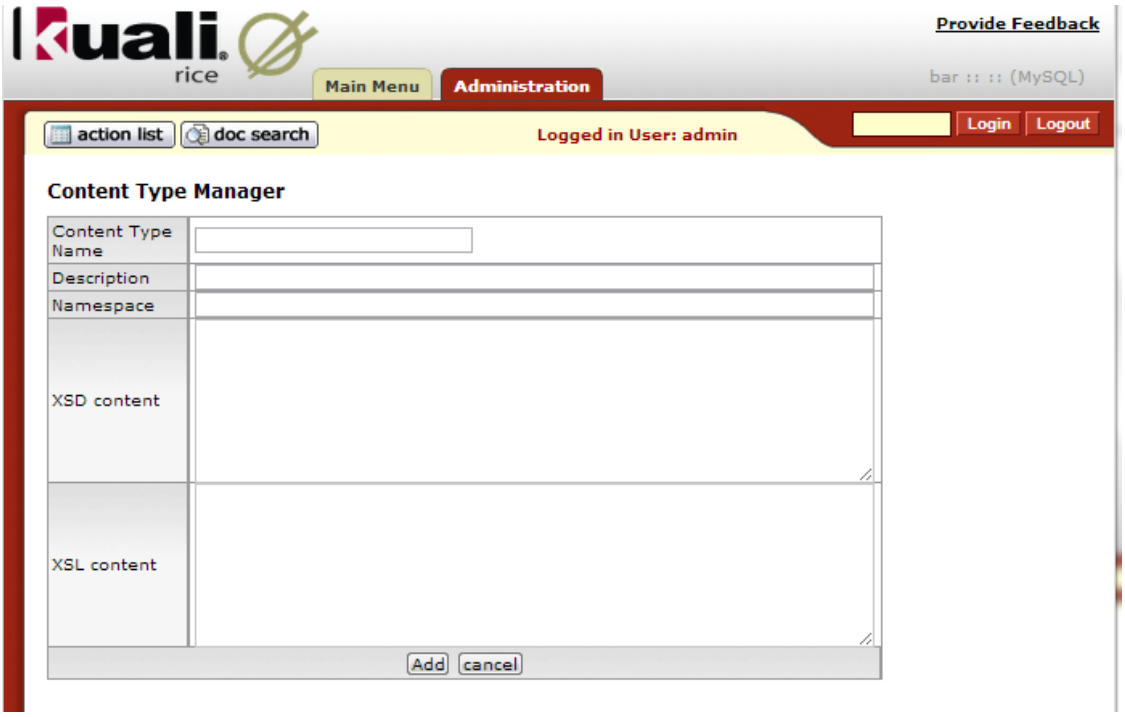
Clicking on the **`Update`** next to the **`Event`** entry brings up the following screen where XSD and XSL content may be updated.

Figure 9.10. Event Content Type



Clicking on the **`Add New Content Type`** above the table brings up the following screen where the new content type, its description, namespace, and its associated XSD and XSL content may be added.

Figure 9.11. New Content Type



Web Service URL

By default, the Notification Web Service API may be accessed at: http://yourlocalip:8080/remoting/soap/ken/v2_0/sendNotificationService

A WSDL may be obtained using the following URL: http://yourlocalip:8080/remoting/soap/ken/v2_0/sendNotificationService?wsdl

Note

In the URLs above, replace yourlocalip with the hostname where KEN is deployed.

Exposed Web Services

Initially, KEN exposes a web service method to send a notification. The *sendNotification* method is a simple String In/String Out method. It accepts one parameter (*notificationMessageAsXml*) and returns a notificationResponse as a String. For the format of the response, see the *Notification Response* section of this document.

Calling the *sendNotification* Service from JAVA

First, create a String that includes the XML content for the notification, as described in the Message Content section of this document. In the following example code, the XML representation of the notification is

read as a file from the file system in the main method, and the code calls the *MySendNotification* method to invoke the Notification web service.

A SOAP style web services binding stub is available in the **notification.jar** file.

You may use this code as a template for sending a notification using the web service:

```
package edu.cornell.library.notification;


import org.apache.commons.io.IOUtils;
import org.kuali.notification.client.ws.stubs.NotificationWebServiceSoapBindingStub;


import java.io.IOException;

import java.io.InputStream;
import java.net.URL;


public class MyNotificationWebServiceClient   {
  private final  static String WEB_SERVICE_URL = "http://localhost:8080/notification/services/Notification";

  public static void MySendNotification(String notificationMessageAsXml) throws Exception {
    URL url = new URL(WEB_SERVICE_URL);
    NotificationWebServiceSoapBindingStub stub = new NotificationWebServiceSoapBindingStub(url, null);
    String responseAsXml = stub.sendNotification(notificationMessageAsXml);
    // do something useful with the response
    System.out.println(responseAsXml);
  }

  public static void main(String[] args) {
    InputStream notificationXML =
 MyNotificationWebServiceClient.class.getResourceAsStream("webservice_notification.xml");
    String notificationMessageAsXml = "";
    try {
      notificationMessageAsXml = IOUtils.toString(notificationXML);
    } catch (IOException ioe) {
      throw new RuntimeException("Error loading webservice_notification.xml");
    }

    try {
      MySendNotification(notificationMessageAsXml);
    } catch (Exception ioe) {
      throw new RuntimeException("Error running webservice");
    }
  }

}
```

Chapter 10. KEN Authentication

Web

KEN can support any Web Sign On technology that results in the population of the HttpServletRequest remote user variable, exposed via the getRemoteUser accessor.

public java.lang.String getRemoteUser()

Returns the login of the user making this request, if the user has been authenticated, or null if the user has not been authenticated. Whether the user name is sent with each subsequent request depends on the browser and type of authentication.

Returns: A *String* specifying the login of the user making this request, or *null*

The generic KEN release comes configured with CAS.

Web Services

Web service authentication is part of the development process and is not implemented by the standalone release of Rice. The notification web service is Axis-based.

Glossary

A

Action List	A list of the user's notification and workflow items. Also called the user's Notification List. Clicking an item in the Action List displays details about that notification, if the item is a notification, or displays that document, if it is a workflow item. The user will usually load the document from their Action List in order to take the requested action against it, such as approving or acknowledging the document.
Action List Type	This tells you if the Action List item is a notification or a more specific workflow request item. When the Action List item is a notification, the Action List Type is "Notification."
Action Request	A request to a user or Workgroup to take action on a document. It designates the type of action that is requested, which includes: • Approve: requests an approve or disapprove action. • Complete: requests a completion of the contents of a document. This action request is displayed in the Action List after the user saves an incomplete document. • Acknowledge: requests an acknowledgment by the user that the document has been opened - the doc will not leave the Action List until acknowledgment has occurred; however, the document routing will not be held up and the document will be permitted to transaction into the processed state if neccessary. • FYI: a notification to the user regarding the document. Documents requesting FYI can be cleared directly from the Action List. Even if a document has FYI requests remaining, it will still be permitted to transition into the FINAL state.
Action Request Hierarchy	Action requests are hierarchical in nature and can have one parent and multiple children.
Action Requested	The action one needs to take on a document; also the type of action that is requested by an Action Request. Actions that may be requested of a user are: • Acknowledge: requests that the users states he or she has reviewed the document. • Approve: requests that the user either Approve or Disapprove a document. • Complete: requests the user to enter additional information in a document so that the content of the document is complete. • FYI: intended to simply makes a user aware of the document.
Action Taken	An action taken on a document by a *Reviewer* in response to an Action Request. The Action Taken may be: • Acknowledged: Reviewer has viewed and acknowledged document. • Approved: Reviewer has approved the action requested on document.

- Blanket Approved: Reviewer has requested a blanket approval up to a specified point in the route path on the document.
- Canceled: Reviewer has canceled the document. The document will not be routed to any more reviewers.
- Cleared FYI: Reviewer has viewed the document and cleared all of his or her pending FYI(s) on this document.
- Completed: Reviewer has completed and supplied all data requested on document.
- Created Document: User has created a document
- Disapproved: Reviewer has disapproved the document. The document will not being routed to any subsequent reviewers for approval. Acknowledge Requests are sent to previous approvers to inform them of the disapproval.
- Logged Document: Reviewer has added a message to the Route Log of the document.
- Moved Document: Reviewer has moved the document either backward or forward in its routing path.
- Returned to Previous Node: Reviewer has returned the document to a previous routing node. When a Reviewer does this, all the actions taken between the current node and the return node are removed and all the pending requests on the document are deactivated.
- Routed Document: Reviewer has submitted the document to the workflow engine for routing.
- Saved: Reviewer has saved the document for later completion and routing.
- Superuser Approved Document: *Superuser* has approved the entire document, any remaining routing is cancelled.
- Superuser Approved Node: Superuser has approved the document through all nodes up to (but not including) a specific node. When the document gets to that node, the normal Action Requests will be created.
- Superuser Approved Request: Superuser has approved a single pending Approve or Complete Action Request. The document then goes to the next routing node.
- Superuser Cancelled: Superuser has canceled the document. A Superuser can cancel a document without a pending Action Request to him/her on the document.
- Superuser Disapproved: Superuser has disapproved the document. A Superuser can disapprove a document without a pending Action Request to him/her on the document.

	• Superuser Returned to Previous Node: Superuser has returned the document to a previous routing node. A Superuser can do this without a pending Action Request to him/her on the document.
Activated	The state of an action request when it is has been sent to a user's Action List.
Activation	The process by which requests appear in a user's Action List
Activation Type	Defines how a route node handles activation of Action Requests. There are two standard activation types: • Sequential: Action Requests are activated one at a time based on routing priority. The next Action Request isn't activated until the previous request is satisfied. • Parallel: All Action Requests at the route node are activated immediately, regardless of priority
Active Indicator	An indicator specifying whether an object in the system is active or not. Used as an alternative to complete removal of an object.
Ad Hoc Routing	A type of routing used to route a document to users or groups that are not in the Routing path for that Document Type. When the Ad Hoc Routing is complete, the routing returns to its normal path.
Annotation	Optional comments added by a *Reviewer* when taking action. Intended to explain or clarify the action taken or to advise subsequent Reviewers.
Approve	A type of workflow action button. Signifies that the document represents a valid business transaction in accordance with institutional needs and policies in the user's judgment. A single document may require approval from several users, at multiple route levels, before it moves to final status.
Approver	The user who approves the document. As a document moves through Workflow, it moves one route level at a time. An Approver operates at a particular route level of the document.
Attachment	The pathname of a related file to attach to a Note. Use the "Browse..." button to open the file dialog, select the file and automatically fill in the pathname.
Attribute Type	Used to strongly type or categorize the values that can be stored for the various attributes in the system (e.g., the value of the arbitrary key/value pairs that can be defined and associated with a given parent object in the system).
Authentication	The act of logging into the system. The Out of the box (OOTB) authenticaton implementation in Rice does not require a password as it is intended for testing puposes only. This is something that must be enabled as part of an implementation. Various authentication solutions exist, such as CAS or Shibboleth, that an implementer may want to use depending on their needs.
Authorization	Authorization is the permissions that an authenticated user has for performing actions in the system.
Author Universal ID	A free-form text field for the full name of the Author of the Note, expressed as "Lastname, Firstname Initial"

B

Base Rule Attribute

The standard fields that are defined and collected for every *Routing Rule* These include:

- Active: A true/false flag to indicate if the Routing Rule is active. If false, then the rule will not be evaluated during routing.
- Document Type: The *Document Type* to which the Routing Rule applies.
- From Date: The inclusive start date from which the Routing Rule will be considered for a match.
- Force Action: a true/false flag to indicate if the review should be forced to take action again for the requests generated by this rule, even if they had taken action on the document previously.
- Name: the name of the rule, this serves as a unique identifier for the rule. If one is not specified when the rule is created, then it will be generated.
- Rule Template: The Rule Template used to create the Routing Rule.
- To Date: The inclusive end date to which the Routing Rule will be considered for a match.

Blanket Approval

Authority that is given to designated *Reviewers* who can approve a document to a chosen route point. A Blanket Approval bypasses approvals that would otherwise be required in the *Routing* For an authorized Reviewer, the *Doc Handler* typically displays the Blanket Approval button along with the other options. When a Blanket Approval is used, the Reviewers who are skipped are sent Acknowledge requests to notify them that they were bypassed.

Blanket Approve Workgroup

A workgroup that has the authority to Blanket Approve a document.

Branch

A path containing one or more Route Nodes that a document traverses during routing. When a document enters a *Split Node* multiple branches can be created. A *Join Node* joins multiple branches together.

Business Rule

1. Describes the operations, definitions and constraints that apply to an organization in achieving its goals.
2. A restriction to a function for a business reason (such as making a specific object code unavailable for a particular type of disbursement). Customizable business rules are controlled by Parameters.

C

Campus

Identifies the different fiscal and physical operating entities of an institution.

Campus Type

Designates a campus as physical only, fiscal only or both.

Cancel

A workflow action available to document initiators on documents that have not yet been routed for approval. Denotes that the document is void and should be disregarded. Canceled documents cannot be modified in any way and do not route for approval.

Canceled	A routing status. The document is denoted as void and should be disregarded.
CAS - Central Authentication Service	http://www.jasig.org/cas - An open source authentication framework. Kuali Rice provides support for integrating with CAS as an authentication provider (among other authentication solutions) and also provides an implementation of a CAS server that integrates with Kuali Identity Management.
Client	A Java Application Program Interface (API) for interfacing with the Kuali Enterprise Workflow Engine.
Client/Server	The use of one computer to request the services of another computer over a network. The workstation in an organization will be used to initiate a business transaction (e.g., a budget transfer). This workstation needs to gather information from a remote database to process the transaction, and will eventually be used to post new or changed information back onto that remote database. The workstation is thus a Client and the remote computer that houses the database is the Server.
Close	A workflow action available on documents in most statuses. Signifies that the user wishes to exit the document. No changes to Action Requests, Route Logs or document status occur as a result of a Close action. If you initiate a document and close it without saving, it is the same as canceling that document.
Comma-separated value	A file format using commas as delimiters utilized in import and export functionality.
Complete	A pending action request to a user to submit a saved document.
Completed	The action taken by a user or group in response to a request in order to finish populating a document with information, as evidenced in the Document Route Log.
Country Restricted Indicator	Field used to indicate if a country is restricted from use in procurement. If there is no value then there is no restriction.
Creation Date	The date on which a document is created.
CSV	See *comma-separated value*

D

Date Approved	The date on which a document was most recently approved.
Date Finalized	The date on which a document enters the FINAL state. At this point, all approvals and acknowledgments are complete for the document.
Deactivation	The process by which requests are removed from a user's *Action List*
Delegate	A user who has been registered to act on behalf of another user. The Delegate acts with the full authority of the Delegator. Delegation may be either *Primary Delegation* or *Secondary Delegation*.
Delegate Action List	A separate Action List for Delegate actions. When a Delegate selects a Delegator for whom to act, an Action List of all documents sent to the Delegator is displayed.

	For both *Primary* and *Secondary Delegation* the Delegate may act on any of the entries with the full authority of the Delegator.
Disapprove	A workflow action that allows a user to indicate that a document does not represent a valid business transaction in that user's judgment. The initiator and previous approvers will receive Acknowledgment requests indicating the document was disapproved.
Disapproved	A status that indicates the document has been disapproved by an approver as a valid transaction and it will not generate the originally intended transaction.
Doc Handler	The Doc Handler is a web interface that a Client uses for the appropriate display of a document. When a user opens a document from the Action List or Document Search, the Doc Handler manages access permissions, content format, and user options according to the requirements of the Client.
Doc Handler URL	The URL for the *Doc Handler*.
Doc Nbr	See *Document Number*.
Document	Also see *E-Doc*. An electronic document containing information for a business transaction that is routed for Actions in KEW. It includes information such as Document ID, Type, Title, Route Status, Initiator, Date Created, etc. In KEW, a document typically has XML content attached to it that is used to make routing decisions.
Document Id	See *Document Number*.
Document Number	A unique, sequential, system-assigned number for a document
Document Operation	A workflow screen that provides an interface for authorized users to manipulate the XML and other data that defines a document in workflow. It allows you to access and open a document by Document ID for the purpose of performing operations on the document.
Document Search	A web interface in which users can search for documents. Users may search by a combination of document properties such as Document Type or Document ID, or by more specialized properties using the Detailed Search. Search results are displayed in a list similar to an Action List.
Document Status	See also *Route Status*.
Document Title	The title given to the document when it was created. Depending on the Document Type, this title may have been assigned by the Initiator or built automatically based on the contents of the document. The Document Title is displayed in both the Action List and Document Search.
Document Type	The Document Type defines the routing definition and other properties for a set of documents. Each document is an instance of a Document Type and conducts the same type of business transaction as other instances of that Document Type. Document Types have the following characteristics: • They are specifications for a document that can be created in KEW

- They contain identifying information as well as policies and other attributes
- They defines the Route Path executed for a document of that type (Process Definition)
- They are hierarchical in nature may be part of a hierarchy of Document Types, each of which inherits certain properties of its *Parent Document Type*.
- They are typically defined in XML, but certain properties can be maintained from a graphical interface

Document Type Hierarchy — A hierarchy of Document Type definitions. Document Types inherit certain attributes from their parent Document Types. This hierarchy is also leveraged by various pieces of the system, including the Rules engine when evaluating rule sets and KIM when evaluating certain Document Type-based permissions.

Document Type Label — The human-readable label assigned to a Document Type.

Document Type Name — The assigned name of the document type. It must be unique.

Document Type Policy — These advise various checks and authorizations for instances of a Document Type during the routing process.

Drilldown — A link that allows a user to access more detailed information about the current data. These links typically take the user through a series of inquiries on different business objects.

Dynamic Node — An advanced type of *Route Node* that can be used to generate complex routing paths on the fly. Typically used whenever the route path of a document cannot be statically defined and must be completely derived from document data.

E

ECL

1. An acronym for Educational Community License.
2. All Kuali software and material is available under the Educational Community License and may be adopted by colleges and universities without licensing fees. The open licensing approach also provides opportunities for support and implementation assistance from commercial affiliates.

E-Doc — An abbreviation for electronic documents, also a shorthand reference to documents created with eDocLite.

eDocLite — A framework for quickly building workflow-enabled documents. Allows you to define document screens in XML and render them using XSL style sheets.

Embedded Client — A type of client that runs an embedded workflow engine.

Employee Status — Found on the Person Document; defines the employee's current employment classification (for example, "A" for Active).

Employee Type — Found on the Person Document; defines the employee's position classification (for example, "P" for Professional).

Entity	An Entity record houses identity information for a given Person, Process, System, etc. Each Entity is categorized by its association with an Entity Type.
Entity Attribute	Entities have directory-like information called Entity Attributes that are associated with them Entity Attributes make up the identity information for an Entity record.
Entity Type	Provides categorization to Entities. For example, a "System" could be considered an Entity Type because something like a batch process may need to interfact with the application.
Exception	A workflow routing status indicating that the document routed to an exception queue because workflow has encountered a system error when trying to process the document.
Exception Messaging	The set of services and configuration options that are responsible for handling messages when they cannot be successfully delivered. Exception Messaging is set up when you configure KSB using the properties outlined in KSB Module Configuration.
Exception Routing	A type of routing used to handle error conditions that occur during the routing of a document. A document goes into Exception Routing when the workflow engine encounters an error or a situation where it cannot proceed, such as a violation of a Document Type Policy or an error contacting external services. When this occurs, the document is routed to the parties responsible for handling these exception cases. This can be a group configured on the document or a responsibility configured in KIM. Once one of these responsible parties has reviewed the situation and approved the document, it will be resubmitted to the workflow engine to attempt the processing again.
Extended Attributes	Custom, table-driven business object attributes that can be established by implementing institutions.
Extension Rule Attribute	One of the rule attributes added in the definition of a rule template that extends beyond the base rule attributes to differentiate the routing rule. A Required Extension Attribute has its "Required" field set to True in the rule template. Otherwise, it is an Optional Extension Attribute. Extension attributes are typically used to add additional fields that can be collected on a rule. They also define the logic for how those fields will be processed during rule evaluation.

F

Field Lookup	The round magnifying glass icon found next to fields throughout the GUI that allow the user to look up reference table information and display (and select from) a list of valid values for that field.
Final	A workflow routing status indicating that the document has been routed and has no pending approval or acknowledgement requests.
Flexible Route Management	A standard KEW routing scheme based on rules rather than dedicated table-based routing.
FlexRM (Flexible Route Module)	The Route Module that performs the Routing for any Routing Rule is defined through FlexRM. FlexRM generates Action Requests when a Rule matches the

	data value contained in a document. An abbreviation of "Flexible Route Module." A standard KEW routing scheme that is based on rules rather than dedicated table-based routing.
Force Action	A true/false flag that indicates if previous Routing for approval will be ignored when an *Action Request* is generated. The flag is used in multiple contexts where requests are generated (e.g., rules, ad hoc routing). If Force Action is False, then prior Actions taken by a user can satisfy newly generated requests. If it is True, then the user needs to take another Action to satisfy the request.
FYI	A workflow action request that can be cleared from a user's Action List with or without opening and viewing the document. A document with no pending approval requests but with pending Acknowledge requests is in Processed status. A document with no pending approval requests but with pending FYI requests is in Final status. See also *Ad Hoc Routing* and *Action Request*.

G

Group	A Group has members that can be either *Principals* or other Groups (nested). Groups essentially become a way to organize Entities (via Principal relationships) and other Groups within logical categories. Groups can be given authorization to perform actions within applications by assigning them as members of *Roles*. Groups can also have arbitrary identity information (i.e., *Group Attributes* hanging from them. Group Attributes might be values for "Office Address," "Group Leader," etc. Groups can be maintained at runtime through a user interface that is capable of workflow.
Group Attribute	Groups have directory-like information called Group Attributes hanging from them. "Group Phone Number" and "Team Leader" are examples of Group Attributes. Group Attributes make up the identity information for a Group record. Group Attributes can be maintained at runtime through a user interface that is capable of workflow.

H

Hierarchical Tree Structure	A hierarchical representation of data in a graphical form.

I

Initialized	The state of an Action Request when it is first created but has not yet been Activated (sent to a user's Action List).
Initiated	A workflow routing status indicating a document has been created but has not yet been saved or routed. A Document Number is automatically assigned by the system.

Initiator	A user role for a person who creates (initiates or authors) a new document for routing. Depending on the permissions associated with the Document Type, only certain users may be able to initiate documents of that type.
Inquiry	A screen that allows a user to view information about a business object.

J

Join Node	The point in the routing path where multiple branches are joined together. A Join Node typically has a corresponding *Split Node* for which it joins the branches.

K

KC - Kuali Coeus	TODO
KCA - Kuali Commercial Affiliates	A designation provided to commercial affiliates who become part of the Kuali Partners Program to provide for-fee guidance, support, implementation, and integration services related to the Kuali software. Affiliates hold no ownership of Kuali intellectual property, but are full KPP participants. Affiliates may provide packaged versions of Kuali that provide value for installation or integration beyond the basic Kuali software. Affiliates may also offer other types of training, documentation, or hosting services.
KCB – Kuali Communications Broker	KCB is logically related to KEN. It handles dispatching messages based on user preferences (email, SMS, etc.).
KEN - Kuali Enterprise Notification	A key component of the Enterprise Integration layer of the architecture framework. Its features include: • Automatic Message Generation and Logging • Message integrity and delivery standards • Delivery of notifications to a user's Action List
KEW – Kuali Enterprise Workflow	Kuali Enterprise Workflow is a general-purpose electronic routing infrastructure, or workflow engine. It manages the creation, routing, and processing of electronic documents (eDocs) necessary to complete a transaction. Other applications can also use Kuali Enterprise Workflow to automate and regulate the approval process for the transactions or documents they create.
KFS – Kuali Financial System	Delivers a comprehensive suite of functionality to serve the financial system needs of all Carnegie-Class institutions. An enhancement of the proven functionality of Indiana University's Financial Information System (FIS), KFS meets GASB and FASB standards while providing a strong control environment to keep pace with advances in both technology and business. Modules include financial transactions, general ledger, chart of accounts, contracts and grants, purchasing/accounts payable, labor distribution, budget, accounts receivable and capital assets.
KIM – Kuali Identity Management	A Kuali Rice module, Kuali Identity Management provides a standard API for persons, groups, roles and permissions that can be implemented by an institution. It also provdies an out of the box reference implementation that allows for a university to use Kuali as their Identity Management solution.

KNS – Kuali Nervous System	A core technical module composed of reusable code components that provide the common, underlying infrastructure code and functionality that any module may employ to perform its functions (for example, creating custom attributes, attaching electronic images, uploading data from desktop applications, lookup/ search routines, and database interaction).
KPP - Kuali Partners Program	The Kuali Partners Program (KPP) is the means for organizations to get involved in the Kuali software community and influence its future through voting rights to determine software development priorities. Membership dues pay staff to perform Quality Assurance (QA) work, release engineering, packaging, documentation, and other work to coordinate the timely enhancement and release of quality software and other services valuable to the members. Partners are also encouraged to tender functional, technical, support or administrative staff members to the Kuali Foundation for specific periods of time.
KRAD - Kuali Rapid Application Development	TODO
KRMS - Kuali Rules Management System	TODO
KS - Kuali Student	Delivers a means to support students and other users with a student-centric system that provides real-time, cost-effective, scalable support to help them identify and achieve their goals while simplifying or eliminating administrative tasks. The high-level entities of person (evolving roles-student, instructor, etc.), time (nested units of time - semesters, terms, classes), learning unit (assigned to any learning activity), learning result (grades, assessments, evaluations), learning plan (intentions, activities, major, degree), and learning resources (instructors, classrooms, equipment). The concierge function is a self-service information sharing system that aligns information with needs and tasks to accomplish goals. The support for integration of locally-developed processes provides flexibility for any institution's needs.
KSB – Kuali Service Bus	Provides an out-of-the-box service architecture and runtime environment for Kuali Applications. It is the cornerstone of the Service Oriented Architecture layer of the architectural reference framework. The Kuali Service Bus consists of: • A services registry and repository for identifying and instantiating services • Run time monitoring of messages • Support for synchronous and asynchronous service and message paradigms
Kuali	1. Pronounced "ku-wah-lee". A partnership organization that produces a suite of community-source, modular administrative software for Carnegie-class higher education institutions. See also *Kuali Foundation* 2. (n.) A humble kitchen wok that plays an important role in a successful kitchen.
Kuali Foundation	Employs staff to coordinate partner efforts and to manage and protect the Foundation's intellectual property. The Kuali Foundation manages a growing portfolio of enterprise software applications for colleges and universities. A lightweight Foundation staff coordinates the activities of Foundation members for critical software development and coordination activities such as source code control, release engineering, packaging, documentation, project management,

	software testing and quality assurance, conference planning, and educating and assisting members of the Kuali Partners program.
Kuali Rice	Provides an enterprise-class middleware suite of integrated products that allow both Kuali and non-Kuali applications to be built in an agile fashion, such that developers are able to react to end-user business requirements in an efficient manner to produce high-quality business applications. Built with Service Oriented Architecture (SOA) concepts in mind, KR enables developers to build robust systems with common enterprise workflow functionality, customizable and configurable user interfaces with a clean and universal look and feel, and general notification features to allow for a consolidated list of work "action items." All of this adds up to providing a re-usable development framework that encourages a simplified approach to developing true business functionality as modular applications.

L

Last Modified Date	The date on which the document was last modified (e.g., the date of the last action taken, the last action request generated, the last status changed, etc.).

M

Maintenance Document	An e-doc used to establish and maintain a table record.
Message	The full description of a *notification message*. This is a specific field that can be filled out as part of the Simple Message or Event Message form. This can also be set by the programmatic interfaces when sending notifications from a client system.
Message Queue	Allows administrators to monitor messages that are flowing through the Service Bus. Messages can be edited, deleted or forwarded to other machines for processing from this screen.

N

Namespace	A Namespace is a way to scope both *Permissions* and *Entity Attributes* Each Namespace instance is one level of scoping and is one record in the system. For example, "KRA" or "KC" or "KFS" could be a Namespace. Or you could further break those up into finer-grained Namespaces such that they would roughly correlate to functional modules within each application. Examples could be "KRA Rolodex", "KC Grants", "KFS Chart of Accounts". Out of the box, the system is bootstrapped with numerous Rice namespaces which correspond to the different modules. There is also a default namespace of "KUALI". Namespaces can be maintained at runtime through a maintenance document.
Note Text	A free-form text field for the text of a Note
Notification Content	This section of a *notification message* which displays the actual full message for the notification along with any other content-type-specific fields.

Notification Message	The overall Notification item or Notification Message that a user sees when she views the details of a notification in her Action List. A Notification Message contains not only common elements such as Sender, Channel, and Title, but also content-type-specific fields.

O

OOTB	Stands for "out of the box" and refers to the base deliverable of a given feature in the system.
Optimistic Locking	A type of "locking" that is placed on a database row by a process to prevent other processes from updating that row before the first process is complete. A characteristic of this locking technique is that another user who wants to make modifications at the same time as another user is permitted to, but the first one who submits their changes will have them applied. Any subsequent changes will result in the user being notified of the optimistic lock and their changes will not be applied. This technique assumes that another update is unlikely.
Optional Rule Extension Attribute	An Extension Attribute that is not required in a Rule Template. It may or may not be present in a *Routing Rule* created from the Template. It can be used as a conditional element to aid in deciding if a Rule matches. These Attributes are simply additional criteria for the Rule matching process.
Org Doc #	The originating document number.
Organization	Refers to a unit within the institution such as department, responsibility center, campus, etc.
Organization Code	Represents a unique identifier assigned to units at many different levels within the institution (for example, department, responsibility center, and campus).

P

Parameter Component Code	Code identifying the parameter Component.
Parameter Description	This field houses the purpose of this parameter.
Parameter Name	This will be used as the identifier for the parameter. Parameter values will be accessed using this field and the namespace as the key.
Parameter Type Code	Code identifying the parameter type. Parameter Type Code is the primary key for its' table.
Parameter Value	This field houses the actual value associated with the parameter.
Parent Document Type	A Document Type from which another *Document Type* derives. The child type can inherit certain properties of the parent type, any of which it may override. A Parent Document Type may have a parent as part of a hierarchy of document types.
Parent Rule	A Routing Rule in KEW from which another Routing Rule derives. The child Rule can inherit certain properties of the parent Rule, any of which it may override. A Parent Rule may have a parent as part of a hierarchy of Rules.
Permission	Permissions represent fine grained actions that can be mapped to functionality within a given system. Permissions are scoped to *Namespace* which roughly correlate to modules or sections of functionality within a given system.

	A developer would code authorization checks in their application against these permissions. Some examples would be: "canSave", "canView", "canEdit", etc. Permissions are aggregated by *Roles*. Permissions can be maintained at runtime through a user interface that is capable of workflow; however, developers still need to code authorization checks against them in their code, once they are set up in the system. Attributes 1. Id - a system generated unique identifier that is the primary key for any Permission record in the system 2. Name - the name of the permission; also a human understandable unique identifier 3. Description - a full description of the purpose of the Permission record 4. Namespace - the reference to the associated *Namespace* Relationships 1. Permission to *Role* - many to many; this relationship ties a Permission record to a Role that is authorized for the Permission 2. Permission to *Namespace* - many to one; this relationship allows for scoping of a Permission to a Namespace that contains functionality which keys its authorization checking off of said
Person Identifier	The username of an individual user who receives the document ad hoc for the Action Requested
Person Role	Creates or maintains the list used in selection of personnel when preparing the Routing Form document.
Pessimistic Locking	A type of lock placed on a database row by a process to prevent other processes from reading or updating that row until the first process is finished. This technique assumes that another update is likely.
Plugins	A plugin is a packaged set of code providing essential services that can be deployed into the Rice standalone server. Plugins usually contains only classes used in routing such as custom rules or searchable attributes, but can contain client application specific services. They are usually used only by clients being implemented by the 'Thin Client' method
Post Processor	A routing component that is notified by the workflow engine about various events pertaining to the routing of a specific document (e.g., node transition, status change, action taken). The implementation of a Post Processor is typically specific to a particular set of Document Types. When all required approvals are completed, the engine notifies the Post Processor accordingly. At this point, the Post Processor is responsible for completing the business transaction in the manner appropriate to its Document Type.

Posted Date/Time Stamp	A free-form text field that identifies the time and date at which the Notes is posted.
Postal Code	Defines zip code to city and state cross-references.
Preferences	User options in an Action List for displaying the list of documents. Users can click the Preferences button in the top margin of the Action List to display the Action List Preferences screen. On the Preferences screen, users may change the columns displayed, the background colors by Route Status, and the number of documents displayed per page.
Primary Delegation	The Delegator turns over full authority to the Delegate. The Action Requests for the Delegator only appear in the Action List of the Primary Delegate. The Delegation must be registered in KEW or KIM to be in effect.
Principal	A Principal represents an *Entity* that can authenticate into the system. One can roughly correlate a Principal to a login username. Entities can exist in KIM without having permissions or authorization to do anything; therefore, a Principal must exist and must be associated with an Entity in order for it to have access privileges. All authorization that is not specific to *Groups* is tied to a Principal. In other words, an Entity is for identity while a Principal is for access management. Also note that an Entity is allowed to have multiple Principals associated with it. The use case typically given here is that a person may apply to a school and receive one log in for the application system; however, once accepted, they may receive their official login, but use the same identity information set up for their Entity record.
Processed	A routing status indicating that the document has no pending approval requests but still has one or more pending acknowledgement requests.

R

Recipient Type	The type of entity that is receiving an Action Request. Can be a user, workgroup, or role.
Required Rule Extension Attribute	An Extension Attribute that is required in a Rule Template. It will be present in every Routing Rule created from the Template.
Responsibility	See *Responsible Party*.
Responsibility Id	A unique identifier representing a particular responsibility on a rule (or from a *route module* This identifier stays the same for a particular responsibility no matter how many times a rule is modified.
Responsible Party	The Reviewer defined on a routing rule that receives requests when the rule is successfully executed. Each routing rule has one or more responsible parties defined.
Reviewer	A user acting on a document in his/her *Action List* and who has received an *Action Request* for the document.
Rice	An abbreviation for Kuali Rice.
Role	Roles aggregate *Permissions* When Roles are given to *Entities* (via their relationship with Principals) or *Groups* an authorization for all associated Permissions is granted.

Route Header Id	Another name for the *Document Id*.
Route Log	Displays information about the routing of a document. The Route Log is usually accessed from either the Action List or a Document Search. It displays general document information about the document and a detailed list of Actions Taken and pending *Action Requests* for the document. The Route Log can be considered an audit trail for a document.
Route Module	A routing component that the engine uses to generate action requests at a particular *Route Node*. *FlexRM* (Flexible Route Module) is a general Route Module that is rule-based. Clients can define their own Route Modules that can conduct specialized Routing based on routing tables or any other desired implementation.
Route Node	Represents a step in the routing process of a document type. Route node "instances" are created dynamically as a document goes through its routing process and can be defined to perform any function. The most common functions are to generate Action Requests or to split or join the route path. • Simple: do some arbitrary work • Requests: generate action requests using a Route Module or the Rules engine • Split: split the route path into one or more parallel branches • Join: join one or more branches back together • Sub Process: execute another route path inline • Dynamic: generate a dynamic route path
Route Path	The path a document follows during the routing process. Consists of a set of route nodes and branches. The route path is defined as part of the *document type* definition.
Route Status	The status of a document in the course of its routing: • Approved: These documents have been approved by all required reviewers and are waiting additional postprocessing. • Cancelled: These documents have been stopped. The document's initiator can 'Cancel' it before routing begins or a reviewer of the document can cancel it after routing begins. When a document is cancelled, routing stops; it is not sent to another Action List. • Disapproved: These documents have been disapproved by at least one reviewer. Routing has stopped for these documents. • Enroute: Routing is in progress on these documents and an action request is waiting for someone to take action. • Exception: A routing exception has occurred on this document. Someone from the Exception Workgroup for this Document Type must take action on this document, and it has been sent to the Action List of this workgroup. • Final: All required approvals and all acknowledgements have been received on these documents. No changes are allowed to a document that is in Final status.

- Initiated: A user or a process has created this document, but it has not yet been routed to anyone's Action List.
- Processed: These documents have been approved by all required users, and is completed on them. They may be waiting for Acknowledgements. No further action is needed on these documents.
- Saved: These documents have been saved for later work. An author (initiator) can save a document before routing begins or a reviewer can save a document before he or she takes action on it. When someone saves a document, the document goes on that person's Action List.

Routed By User

The user who submits the document into routing. This is often the same as the Initiator. However, for some types of documents they may be different.

Routing

The process of moving a document through its route path as defined in its Document Type. Routing is executed and administered by the workflow engine. This process will typically include generating Action Requests and processing actions from the users who receive those requests. In addition, the Routing process includes callbacks to the Post Processor when there are changes in document state.

Routing Priority

A number that indicates the routing priority; a smaller number has a higher routing priority. Routing priority is used to determine the order that requests are activated on a route node with sequential activation type.

Routing Rule

A record that contains the data for the *Rule Attributes* specified in a *Rule Template* It is an instance of a Rule Template populated to determine the appropriate Routing. The Rule includes the Base Attributes, Required Extension Attributes, Responsible Party Attributes, and any Optional Extension Attributes that are declared in the Rule Template. Rules are evaluated at certain points in the routing process and, when they fire, can generate Action Requests to the responsible parties that are defined on them.

Technical considerations for a Routing Rules are:

- Configured via a GUI (or imported from XML)
- Created against a Rule Template and a Document Type
- The Rule Template and it's list of Rule Attributes define what fields will be collected in the Rule GUI
- Rules define the users, groups and/or roles who should receive action requests
- Available Action Request Types that Rules can route
  - Complete
  - Approve
  - Acknowledge
  - FYI
- During routing, Rule Evaluation Sets are "selected" at each node. Default is to select by Document Type and Rule Template defined on the Route Node

- Rules match (or 'fire') based on the evaluation of data on the document and data contained on the individual rule
- Examples
  - If dollar amount is greater than $10,000 then send an Approval request to Joe.
  - If department is "HR" request an Acknowledgment from the HR.Acknowledgers workgroup.

Rule Attribute

Rule attributes are a core KEW data element contained in a document that controls its Routing. It participates in routing as part of a Rule Template and is responsible for defining custom fields that can be rendered on a routing rule. It also defines the logic for how rules that contain the attribute data are evaluated.

Technical considerations for a Rule Attribute are:

- They might be backed by a Java class to provide lookups and validations of appropriate values.
- Define how a Routing Rule evaluates document data to determine whether or not the rule data matches the document data.
- Define what data is collected on a rule.
- An attribute typically corresponds to one piece of data on a document (i.e dollar amount, department, organization, account, etc.).
- Can be written in Java or defined using XML (with matching done by XPath).
- Can have multiple GUI fields defined in a single attribute.

Rule QuickLinks

A list of document groups with their document hierarchies and actions that can be selected. For specific document types, you can create the rule delegate.

Rule Template

A Rule Template serves as a pattern or design for the routing rules. All of the Rule Attributes that include both Required and _Optional_ are contained in the Rule Template; it defines the structure of the routing rule of FlexRM. The Rule Template is also used to associate certain Route Nodes on a document type to routing rules.

Technical considerations for a Rule Templates are:

- They are a composition of Rule Attributes
- Adding a 'Role' attribute to a template allows for the use of the Role on any rules created against the template
- When rule attributes are used for matching on rules, each attribute is associated with the other attributes on the template using an implicit 'and' logic attributes
- Can be used to define various other aspects to be used by the rule creation GUI such as rule data defaults (effective dates, ignore previous, available request types, etc)

S

Save	A workflow action button that allows the Initiator of a document to save their work and close the document. The document may be retrieved from the initiator's Action List for completion and routing at a later time.
Saved	A routing status indicating the document has been started but not yet completed or routed. The Save action allows the initiator of a document to save their work and close the document. The document may be retrieved from the initiator's action list for completion and routing at a later time.
Searchable Attributes	Attributes that can be defined to index certain pieces of data on a document so that it can be searched from the *Document Search screen*. Technical considerations for a Searchable Attributes are: • They are responsible for extracting and indexing document data for searching • They allow for custom fields to be added to Document Search for documents of a particular type • They are configured as an attribute of a Document Type • They can be written in Java or defined in XML by using Xpath to facilitate matching
Secondary Delegation	The Secondary Delegate acts as a temporary backup Delegator who acts with the same authority as the primary Approver/the Delegator when the Delegator is not available. Documents appear in the Action Lists of both the Delegator and the Delegate. When either acts on the document, it disappears from both Action Lists. Secondary Delegation is often configured for a range of dates and it must be registered in KEW or KIM to be in effect.
Service Registry	Displays a read-only view of all of the services that are exposed on the Service Bus and includes information about them (for example, IP Address, or Endpoint URL).
Simple Node	A type of node that can perform any function desired by the implementer. An example implementation of a simple node is the node that generates Action Requests from route modules.
SOA	An acronym for Service Oriented Architecture.
Special Condition Routing	This is a generic term for additional route levels that might be triggered by various attributes of a transaction. They can be based on the type of document, attributes of the accounts being used, or other attributes of the transaction. They often represent special administrative approvals that may be required.
Split Node	A node in the routing path that can split the route path into multiple branches.
Spring	The Spring Framework is an open source application framework for the Java platform.
State	Defines U.S. Postal Service codes used to identify states.
Status	On an Action List; also known as Route Status. The current location of the document in its routing path.

Submit	A workflow action button used by the initiator of a document to begin workflow routing for that transaction. It moves the document (through workflow) to the next level of approval. Once a document is submitted, it remains in 'ENROUTE' status until all approvals have taken place.
Superuser	A user who has been given special permission to perform Superuser Approvals and other Superuser actions on documents of a certain Document Type.
Superuser Approval	Authority given Superusers to approve a document of a chosen Route Node. A Superuser Approval action bypasses approvals that would otherwise be required in the Routing. It is available in Superuser Document Search. In most cases, reviewers who are skipped are not sent Acknowledge Action Requests.
Superuser Document Search	A special mode of Document Search that allows Superusers to access documents in a special Superuser mode and perform administrative functions on those documents. Access to these documents is governed by the user's membership in the Superuser Workgroup as defined on a particular Document Type.

T

Thread pool	A technique that improves overall system performance by creating a pool of threads to execute multiple tasks at the same time. A task can execute immediately if a thread in the pool is available or else the task waits for a thread to become available from the pool before executing.
Title	A short summary of the notification message. This field can be filled out as part of the Simple Message or Event Message form. In addition, this can be set by the programmatic interfaces when sending notifications from a client system. This field is equivalent to the "Subject" field in an email.

U

URL	An acronym for Uniform Resource Locator.
User	A person who can log in and use the application. This term is synonymous with "Principal" in KIM. "Whereas Entity Id represents a unique Person, Principal Id represents a set of login information for that Person."

V

Viewer	A user(s) who views a document during the routing process. This includes users who have action requests generated to them on a document.

W

Web Service Client	A type of client that connects to a standalone KEW server using Web Services.
Wildcard	A character that may be substituted for any of a defined subset of all possible characters.
Workflow	Electronic document routing, approval and tracking. Also known as Workflow Services or Kuali Enterprise Workflow (KEW). The Kuali infrastructure service

	that electronically routes an e-doc to its approvers in a prescribed sequence, according to established business rules based on the e-doc content. See also *Kuali Enterprise Workflow*.
Workflow Engine	The component of KEW that handles initiating and executing the route path of a document.
Workflow QuickLinks	A web interface that provides quick navigation to various functions in KEW. These include: • Quick EDoc Watch: The last five Actions taken by this user. The user can select and repeat these actions. • Quick EDoc Search: The last five EDocs searched for by this user. The user can select one and repeat that search. • Quick Action List: The last five document types the user took action with. The user can select one and repeat that action.

X

XML	See also *XML Ingester*. 1. An acronym for Extensible Markup Language. 2. Used for data import/export.
XML Ingester	A workflow function that allows you to browse for and upload XML data.
XML RuleAttribute	Similar in functionality to a RuleAttribute but built using XML only