|  |  |
| --- | --- |
|  | **KD_logo_159box**  **KINETIC REQUEST** Notification Console Implementation  **January 5, 2015** |

**Version: 1.0**

**Author: Anne Ramey (Kinetic Data)**

# 

# tABLE OF CONTENTS

[tABLE OF CONTENTS 2](#_Toc441489874)

[Project Overview 2](#_Toc441489875)

[Version Control Chart 2](#_Toc441489876)

[Authentication AND PERMISSIONS 3](#_Toc441489877)

[PACKAGE 4](#_Toc441489878)

[Notification Console Specific CONFIGURATION 4](#_Toc441489879)

[Notification Console 5](#_Toc441489880)

[Using the Notifications 7](#_Toc441489881)

[Notification Subtree/Routine 7](#_Toc441489882)

[Appearance/Format Processing 8](#_Toc441489883)

# 

# Project Overview

Kinetic Data worked to implement the Notification Console to allow for configurable, customizable notification templates that support different emails by region, multi-language, etc.

## Version Control Chart

| Version | Primary Author(s) and Contributors | Description of Version | Date Completed |
| --- | --- | --- | --- |
| 1.0 | Anne Ramey | Initial version | 1/5/2016 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Authentication AND PERMISSIONS

Authentication into the Notification console is controlled the same way as authentication into the rest of the Service Catalog. Permissions on the Notification Console are set as follows:



No permissions are necessary to use the templates within the trees. Developers can use the Notifications set up within the console without having any access to the console itself.

# PACKAGE

The Notification Console package is a package for the Responsive v2 bundle and can be found at packages\notification\_console in the bundle.

# Notification Console Specific CONFIGURATION

The Notification Console has a configuration file config/config.jspf that contains a set of properties for the package. These are configurations meant to be changed for each implementation.

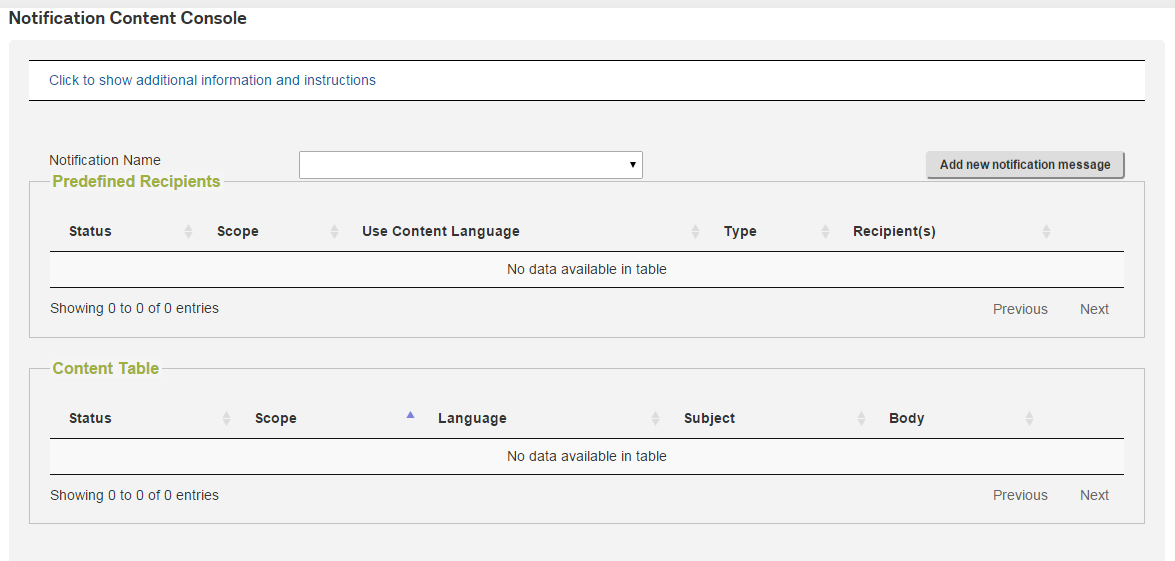
* basePackageUrl

Define url to package page.

bundle.getProperty("displayPageUrlSlug") + "notificationsPackage"

There is another configuration file, config\config.json that is handles how the package drops in and integrates with the responsive bundle. This file defines the link and icon available in the parent bundle, and the view(s), the callbacks available.

# Notification Console

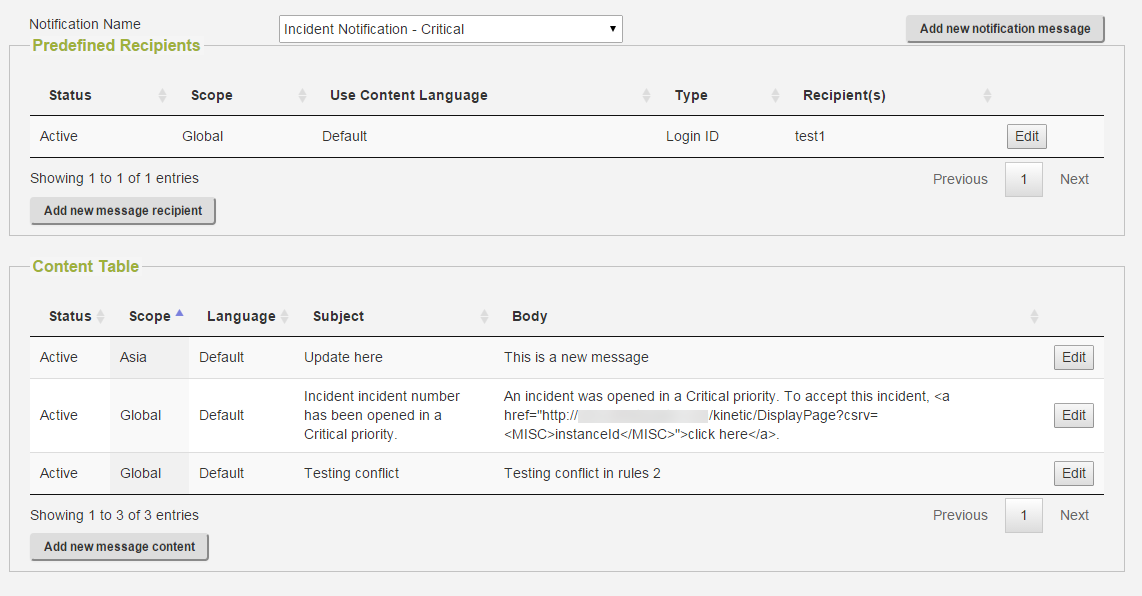


When the console first displays (Notification Content Console service item), it displays the default view (interface/views/package/package.jsp) using the package css (assets\css\notification.css) with no additional calls made because no selections are defaulted.

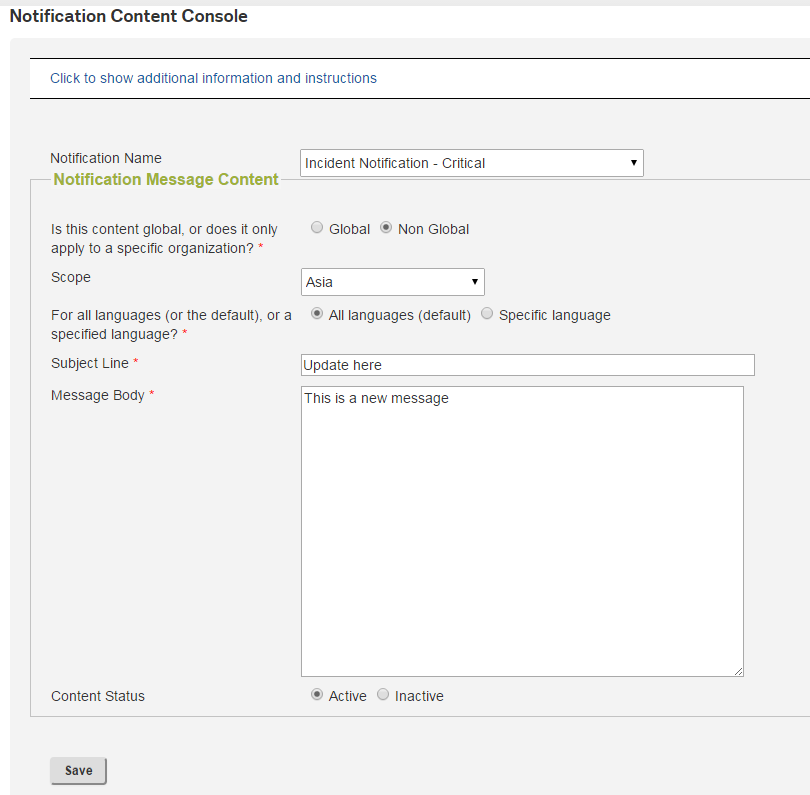
Additional Information and detailed instructions on how to use/build notifications are available by clicking on “Click to show additional information and instructions”. Once displayed, clicking on the information then hides it again.

To view/edit the settings/content for an existing notification, select a notification from the Notification Name drop down, which is the \_Notification Name dynamic list question off the service item. This uses the package js (assets\js\notifications.js) to load the data for that notification into the tables on the screen using bridge queries.

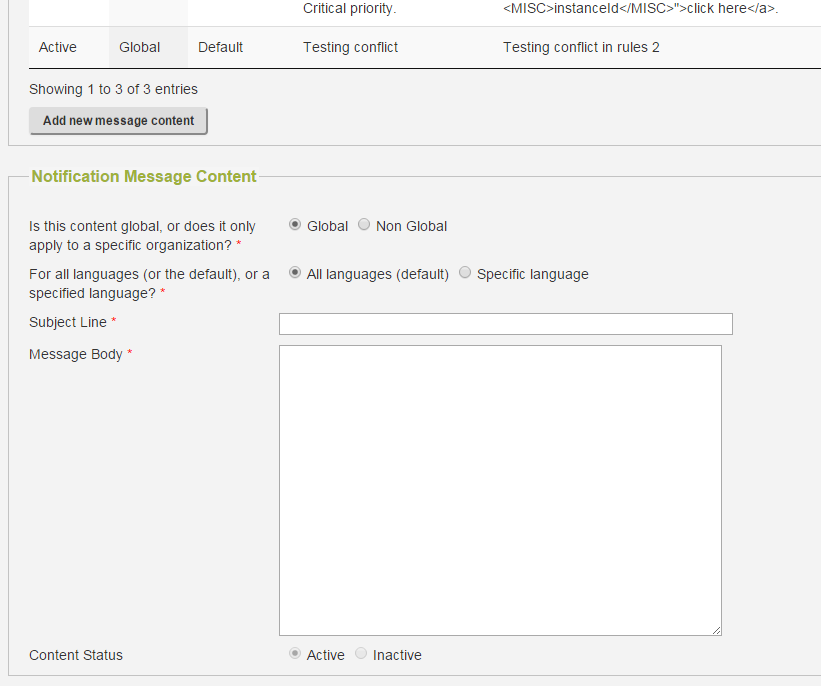
Any existing entry has an “Edit” button, and the tables each have an “Add new” button as well.



Once you select to edit an item, that item displays and the tables disappear:



But new will display the existing items above the new entry:



For both the save and the update, the updates are handled by the javascript (assets\js\notifications.js) and either the create or update notification callback (interface/callbacks/package/createNotification.jsp or interface/callbacks/package/updateNotification.jsp).

Adding a whole new notification is done with the button in the upper left, which will expose a couple different sections on the service item, asking for the new notification name and the message recipient(s), then for the initial message content. This add record is handled the same way as the add for an existing message, though, in the files. Everything is handled in assets\js\notifications.js using interface/callbacks/package/createNotification.jsp.

# Using the Notifications

The notifications are sent using a subtree/routine called Send Template Email Notification. So, to use the notification(s), the developer uses this tree/routine in their tree. It’s a matter of dropping in the node and filling in the parameters, as with any handler.

The parameters are as follows:

|  |  |
| --- | --- |
| Parameter | Information/description |
| Email Addresses To Email | List of comma delimited email addresses to email |
| Login IDs To Email | List of comma delimited login ids to email (email addresses looked up in CTM:People) |
| Email Template Name | (Required) Notification name from Notification Console |
| Format Answers |  |
| JSON Misc. | JSON input of miscellaneous values (if any are needed for this notification) |
| Request Instance ID | instanceId of the Kinetic Request submission to get answers/dataset for, recommended if referencing answers or dataset in notifications |
| Region | Region to send the message to/for, recommended if using predefined recipients |
| Language | Language to send the message in, recommended if using predefined recipients |
| Catalog | (Required) Catalog message is being sent from |

There are no outputs from this subtree/routine.

# Notification Subtree/Routine

The notification subtree/routine requires the following handlers:

* Kinetic Request Answers All Retrieve
* Kinetic Request Submission Base Retrieve
* Kinetic Request Submission Dataset Retrieve
* Utility JSON to Results
* Utility CSV to XML
* Kinetic Request Service Item Attribute Retrieve
* Helper Form Notification Content Lookup
* Helper Form Language Preference Lookup
* BMC ITSM7 Person Retrieve
* SMTP Email Send
* Kinetic Request Attachment Retrieve
* SMTP Email with Attachment Send

This tree first retrieves all the information it will need to perform its needed functions, the MISC data, and, if the request ID as provided, the answers, base, and dataset information. Then the tree splits into two paths, one for the predefined recipients and one for the provided recipients. They are not mutually exclusive. If there are both, it will do both.

For the predefined recipients, the base notification is looked up first, then the recipients for that notification. For dynamically defined recipients, the language must be looked up for each, so we look up the people first, then the content within the loop of people.

Note that since attachments must be sent with a different SMTP send handler, there is a different path if there is an attachment to handle along each path.

# Appearance/Format Processing

As part of the notification processing, there may be special elements that need particular formatting beyond just that of language. For example, the date of 1/25/2016 can be specified in any number of ways, including Monday January 25, 2016 or January 25, 2016 or 25-1-2016 or 1/25/2016. All of these would be valid, so which should be used in the email when such a date is passed in? That may be different for different sorts of email communications. Because of this, there is a subtree that handles Appearance Processing and handles date formats. It could be expanded to handle other kinds of formatting as necessary. This is called after the notification base content is found and returns the actual content of the subject and body to the main routine.

The date records can be looked up with the hander:

* Helper Form Notification Format Lookup

With the format type of

Dates

And the record would be stored in the helper form with an Index Field1 of Nofication, an Index Field2 of Formats, and an Index Field3 of Dates. Character Field1 should be the name of the format as referenced in the message with the tags <FORMAT></FORMAT>, and Character Field2 should be a Ruby Date format, like %m/%d/%Y for 1/25/2016, see <http://foragoodstrftime.com/> for a good reference.

If the Data Management Console is installed, it is advisable to use a dataset to manage these date formats.