Kuali Rice Version Compatibility Statement

Overview

As more projects begin to use Kuali Rice as their middleware framework, there is an increasing need to provide compatibility between the different versions of the Kuali Rice software.

In the following document, we will outline what compatibility means for the Kuali Rice software and the applications that use it. Additionally, we will make statements as to the level of compatibility that should be provided for within Kuali Rice. Subsequent to that we will outline a roadmap for achieving these compatibility goals.

It is important to note that the statements made in this document reflect goals for where the Kuali Rice project needs to be in regards to compatibility. As of Kuali Rice version 1.0, it is not the case that we will be able to achieve all of these goals for compatibility between applications using version 1.0 and version 1.1

Definition of Compatibility

There are two different types of version compatibility that need to be addressed in regards to Kuali Rice:

1. Client-Server (Middleware) Version Compatibility
2. Framework and API Compatibility

Client-Server Version Compatibility

In a typical enterprise deployment of Kuali Rice, a standalone Rice server will be deployed which hosts numerous shared services. Other Rice client applications will then be configured to interact with the service hosted by the standalone server.

Client-Server version compatibility is concerned with the compatibility of a Rice client application running a different version of the software than the Rice standalone server it is interacting with.

Framework and API Version Compatibility

In addition to the shared services that are provided by Kuali Rice, there is also a set of components which constitute an application framework. This consists of the APIs, Libraries and web framework which are used to construct a Kuali Rice application.

Framework version compatibility is concerned with the ability of code developed under one version of Rice to be executed under a different version of Rice. This relates to the amount of work required to upgrade a Kuali Rice application from one version of Rice to another.

Note that the standalone Rice server is considered to be a Kuali Rice application since it uses the Rice framework pieces internally.

Versioning of Kuali Rice Software

Versions of Kuali Rice follow a numerical scheme consisting of three parts separated by periods:

- *major.minor.patch*

Major versions

Major versions of Rice are intended to be long-lived versions of the Kuali Rice software which consist of numerous minor versions. The decision to create a new major version of Kuali Rice will be the result of the need for major changes which cannot be successfully implemented without breaking version compatibility.

Therefore it is permitted that different major versions of Kuali Rice will not be compatible with each other.

Minor versions

Most work on the Kuali Rice project will be done as part of a new minor version of the software. It is intended that a high level of compatibility (as described in #Kuali Rice Version Compatibility Statements below) will be be offered between different minor versions within the same major version.

Patch versions

Patch versions should be created to address security issues or bugs with an existing minor version release. It is intended that different patch versions of Kuali Rice within the same minor version will be fully compatible and interoperable with each other.

Kuali Rice Version Compatibility Statements

What follows are statements which indicate the level of compatibility that will be provided between different versions of the Kuali Rice software.

Client-Server Version Compatibility Statement

> *A client application using a particular version of Kuali Rice should be compatible with any Kuali Rice Standalone Server running the same or greater version provided that the server version is within the same major version as the client.*

So, for example, if a client application is at Kuali Rice version 1.1.0, then it should be compatible with any 1.x server version that is greater than or equal to 1.1.0. However, a 1.1.0 client application would not necessarily be compatible with a 2.0.0 server.

Framework and API Compatibility Statement

> *A client application developed using a particular version of Kuali Rice should be able to upgrade to a newer version of Kuali Rice (within the same major version) without forcing changes to be made to application code. However, the following courses of action may be required:*
>
> 1. *Update versions of Rice libraries and dependent libraries*
> 2. *Addition of new libraries*
> 3. *Execution of database conversion scripts*
> 4. *Recompilation of application code*
> 5. *Recompilation and re-packaging of plug-ins*
> 6. *Configuration changes*
>
> *Any of these changes which are required will be documented in the release notes for the new Kuali Rice version.*
>
> *In order for a client application to benefit from this level of API compatibility, certain best practices need to be followed:*
>
> 1. *Use only service and framework pieces from the **api** module of Kuali Rice*
> 2. *Use only supported configuration methods*
> 3. *Use only supported client-server integration methods*
>
> *In the case of certain institutional customizations to the Rice standalone server and services, implementors may use or extend code which falls into the **impl** module of Kuali Rice. In those cases, there may be some impacting work required as part of the upgrade process.*
>
> *Additionally, in rare cases it may be required to make an impacting change to the API between minor versions. In these cases, the changes which are required should be thoroughly documented in the release notes for the new Kuali Rice version.*

For example, a client application which takes advantage of the standard APIs for Kuali Rice should be able to easily upgrade to a new version of Kuali Rice without requiring changes to their application code. However, they will need to get new versions of the Kuali Rice libraries as part of the upgrade process (and any dependent libraries) and recompile against these new libraries.