RSuite Edit Setup

Notice

Notices

Trademarks

RSuite is a registered trademark and the RSuite CMS logo is a tradmark of Orbis Technologies, Inc.

CKSource and CKEditor are trademarks of CKSource.

Chrome and Google Chrome are trademarks and Google is a registered trademark of Google, Inc.

Firefox and Mozilla are registered trademarks of the Mozilla Foundation.

Internet Explorer is a registered trademark and Microsoft is a trademark of Microsoft Corporation.

About RSuite

This is a confidential document to be shared only with current RSuite customers and others designated by Orbis Technologies, Inc.

For more information on RSuite CMS or Orbis Technologies, Inc. visit these web sites:

- Orbis Technologies, Inc. (*www.orbistechnologies.com*)
- RSuite CMS Community (*https://support.rsuitecms.com/support/home*)

The PDF version of this document is created with the RenderX XEP FO engine integrated with the DITA Open Toolkit. See *www.renderx.com* for product details.

Contents

Preface

About this book

RSuite® CMS is a robust content management application, especially suited to loading, storing, managing, transforming, and exporting XML documents within the context of a publishing enterprise.

This book contains instructions for installing RSuite Edit, a WYSIWYG editor with which you can edit DITA files directly from the RSuite UI. RSuite Edit is RSuite's integration of CKEditor™, a product of CKSource™.

The audience for this book is RSuite administrators and developers. The procedures assume the ability to create an RSuite plug-in and to code an extension that adds function to RSuite and RSuite Editor, for instance adding an action to the context menus associated with specific types of managed objects.

Chapter 1

Getting started

Overview

Setting up RSuite® Edit has two main steps: installing the RSuite Edit plug-in, and creating and installing a custom plug-in. The custom plug-in is necessary for assigning the context menu actions that invoke the editor. The plug-in can also contain custom transforms that convert DITA mark-up to XHTML mark-up for editing, and then save the edited files back in RSuite in DITA format.

The principal set-up steps, as described in *Installing the RSuite Edit plug-in* on page 11 and *Configuring the customer plug-in* on page 13, are:

1. Download the RSuite Edit plug-in.
2. Install the RSuite Edit plug-in.
3. Define a context menu rule set to assign the RSuite Edit action to specific types of objects. The instructions include an example you can modify.
4. If you are not going to use the built-in transforms for converting files between DITA and XHTML, write custom transforms.
5. Create the customer plug-in with the context menu rules and any custom transforms.
6. Install the customer plug-in.

Before you begin

Perform these steps to prepare to set up RSuite Edit. You will refer to these decisions during the setup.

1. Make a note of which types of managed objects you will edit with RSuite Edit.

   _____
   _____
   _____
   _____
   _____
   _____

2. Determine whether you will use custom routines for converting files between DITA and XHTML (as opposed to using the transforms that are delivered in the RSuite Edit plug-in). Yes____ No_____
3. Be sure you have the latest version of this set-up manual, and report any browser-related problems to RSuite Support (*https://support.rsuitecms.com/support/home*).

   RSuite Edit has been tested with these browsers and versions:

   Mozilla® Firefox® 50-55
   Google Chrome® 55-59
   Microsoft™ Internet Explorer® 10-11

and should work with any current version.

Chapter 2

Installing the RSuite Edit plug-in

Perform these steps to install the RSuite Edit plug-in.

1. Download the RSuite Edit plug-in from the RSuite Support site (*https://support.rsuitecms.com/support/solutions/articles/9000068195-rsuite-5-plug-ins*).
2. Copy the rsuite-edit-plugin-*v.r*.jar to the `rsuite-home/plugins` folder.

Chapter 3

Configuring the customer plug-in

Topics:

Since customer environments vary and RSuite CMS is highly customizable, the RSuite Edit plug-in does not assign context menu actions. They must be assigned by a customer plug-in. The customer plug-in can also be used to deliver custom transforms and CKEditor plug-ins.

Configuring the RSuite context menu

Use an extension point in an RSuite plug-in to add actions to RSuite context menus.

```
<extensionProvider id="rsuite.ContextMenu">
  <contextMenuRuleSet name="rsuite_edit.contextmenuitem.DITA.topic.all.users"
       scope="allNodes">
    <menuItemList>
      <menuItem id="rsuite_edit:openEditor">
        <actionName>pages:editorBasic</actionName>
        <label>RSuite Edit</label>
        <property name="rsuite:group" value="rsuite:content" />
        <property name="rsuite:icon" value="edit"/>
      </menuItem>
    </menuItemList>
    <ruleList>
      <rule>include nodeType mo</rule>
    </ruleList>
  </contextMenuRuleSet>
</extensionProvider>
```

Figure 1: Sample extension provider for context menus

1. Review the list of managed object types whose context menus should include the RSuite Edit action. You made the list in step *1* on page 9.
2. Copy the sample extension provider and modify it to make RSuite Edit available for the selected file types.

   The sample extension provider puts "RSuite Edit" on the context menu of every type of managed object.
3. Save the customized extension provider in the rsuite-plugin.xml file of the customer plug-in.

Configuring custom transforms

RSuite Edit uses transforms when it checks out (opens) and checks in (saves) content. The RSuite Edit plug-in delivers DITA-to-XHTML and XHTML-to-DITA transforms, but you can override either one, or both, with a custom transform in the customer plug-in.

The built-in transforms are in the WebContent folder of the RSuite Edit plug-in: `../WebContent/xslt/rsuiteEditDita2xhtml/rsuiteEditDita2xhtml.xsl` and `../WebContent/xslt/rsuiteEditXhtml2dita/rsuiteEditXhtml2dita.xsl`. The default tag mappings that they implement are listed at *RSuite Edit built-in transforms* on page 19.

If in step *2* on page 9 you decided to use custom transforms, perform these steps to add your custom transforms to the customer RSuite plug-in.

1. When you create the custom transforms:

   The custom transform file overrides the complete built-in transform file.

   a) To keep built-in mappings, you include them in your custom file.
   b) Give the DITA-to-XHTML transform a name like *mypurpose*dita2xhtml.xsl.
   c) Place it in the WebContent folder of the customer plug-in: `WebContent\xslt\customdita2xhtml\`*`mypurpose`*`dita2xhtml.xsl`.
   d) Give the XHTML-to-DITA transform a name like *mypurpose*xhtml2dita.xsl and place it in the WebContent folder of the customer plug-in: `WebContent\xslt\customxhtml2dita\`*`mypurpose`*`xhtml2dita.xsl`.
2. To implement the custom transforms and override the built-in transforms, declare the locations of the the custom transforms by adding elements to the rsuite.WebEditing extension in the customer plug-in's rsuite-plugin.xml file.

a) To override the built-in DITA-to-XHTML transform, add:

```
<transformToHtml>@pluginId@\xslt\customdita2xhtml
\mypurposeDita2Xhtml.xsl</transformToHtml>
```

b) To override the built XHTML-to-DITA transform, add:

```
<transformToXml>@pluginId@\xslt\customxhtml2dita
\mypurposeXhtml2Dita.xsl</transformToXml>
```

"**@pluginId@**" is a standard token; RSuite resolves it to the WebContent folder of the plug-in.

It should look like this:

```
  <extensionProvider id="rsuite.WebEditing">
     <rsuiteEditConfig>
       <editInstanceConfig>
   …
        <transformToHtml>@pluginId@\xslt\customDita2Xhtml
\mypurposeDita2Xhtml.xsl</transformToHtml>
        <transformToXml>@pluginId@\xslt\customXhtml2Dita
\mypurposeXhtml2Dita.xsl</transformToXml>
   …
       </editInstanceConfig>
     </rsuiteEditConfig>
   </extensionProvider>
```

Adding CKEditor plug-ins

Perform these steps to include a CKEditor plug-in in the customer RSuite plug-in.

Add a downloaded CKEditor plug-in

1. Open the **CKEditor Add-ons List** of popular plug-ins (*https://ckeditor.com/addons/plugins/all*).
2. Download the add-on and unzip it.
3. Package the unzipped add-on in the `WebContent\editor\ckeditorPlugins` folder of your RSuite plug-in.

Add a custom CKEditor plug-in

To create your own CKEditor add-on, you can modify an existing plug-in or create a new plug-in. Perform these steps to create a new plug-in.

1. Open the **Creating a CKEditor Plugin in 20 Lines of Code** page (*https://docs.ckeditor.com/#!/guide/plugin_sdk_sample*).
2. Follow the tutorial at **Developing a Custom Plugin** to create a new add-on.
3. Package the add-on in the `WebContent\editor\ckeditorPlugins` of the your RSuite plug-in

Changing CKEditor properties

CKEditor properties can be changed in the rsuite.WebEditing extension point in a customer RSuite plug-in. There is a list of CKEditor configuration properties on the CKEditor Web site (*https://docs.ckeditor.com/#!/api/CKEDITOR.config*).

Property values are changed with <PROPERTY> elements grouped within a <PROPERTYMAP> element.

```
<extensionProvider id="rsuite.WebEditing">
  <rsuiteEditConfig>
    <editInstanceConfig>
      <propertyMap>
        <property><name>colorButton_colorsPerRow</name><value>8</value></
property>
        <property><name>language</name><value>es</value></property>
        <property><name>codeSnippet_languages</name>
            <value>{javascript: 'JavaScript', php: 'PHP'}</value></property>
        <property><name>scayt_multiLanguageMode</name><value>false</value></
property>
      </propertyMap>
    </editInstanceConfig>
  </rsuiteEditConfig>
</extensionProvider>
```

Properties can have various data types.

String

```
<property><name>language</name><value>es</value></property>
```

Number

```
<property><name>colorButton_colorsPerRow</name><value>8</value></property>
```

JSON

```
<property><name>codeSnippet_languages</name>
      <value>{javascript: 'JavaScript', php: 'PHP'}</value></property>
```

Array

```
<property><name>blockedKeystrokes</name><value>[
          CKEDITOR.CTRL + 66,
          CKEDITOR.CTRL + 73,
          CKEDITOR.CTRL + 85
          ]</value></property>
```

Boolean

```
<property><name>browserContextMenuOnCtrl</name><value>false</value></
property>
```

Appendix
A

RSuite Edit default configuration

Topics:

This appendix describes the initial CKEditor settings delivered by the RSuite Edit plug-in.

RSuite Edit defaults

The RSuite Edit plug-in installs RSuite Edit with the CKEditor defaults except for the settings shown in *Figure 2: RSuite Edit defaults changed from CKEditor* on page 18. CKEditor defaults are documented on the *CKEditor Web site*.

```
config.defaultLanguage = "en";
config.language = RSICKEDITOR.getLang("cmsLocale");
config.languages = ['es:Spanish', 'en:English'];

/**we have full translations for these two languages */
config.extraAllowedContent="div[*]{*}(*);ul[*]{*}(*);img[*]{*}(*);ol[*]{*}
(*);span[*]{*}(*);
td[*]{*}(*); p[*]{*}(*);li[*]{*}(*);b[*]{*}(*);i[*]{*}(*);em[*]{*}
(*);strong[*]{*}(*);
sup[*]{*}(*);sub[*]{*}(*);a[*]{*}(*);h1[*]{*}(*);h2[*]{*}(*);h3[*]{*}
(*);h4[*]{*}(*);
h5[*]{*}(*);h6[*]{*}(*)";
config.removePlugins += 'image, horizontalrule, about';
config.removeButtons = 'horizontalrule, rsiannotationComment, lineutils,
 lite, widget';
config.removeDialogTabs = 'image:advanced;link:advanced';
config.toolbarGroups = [
        {name: 'clipboard', groups: ['clipboard', 'undo']},
        {name: 'editing', groups: ['find', 'selection', 'spellchecker',
 'editing']},
        {name: 'links', groups: ['links']},
        {name: 'insert', groups: ['insert']},
        {name: 'rsi', groups: ['rsi']},
        {name: 'forms', groups: ['forms']},
        {name: 'tools', groups: ['tools']},
        {name: 'document', groups: ['document']},
        {name: 'others', groups: ['others']},
        '/',
        {name: 'basicstyles', groups: ['basicstyles', 'cleanup']},
        {name: 'paragraph', groups: ['list', 'indent', 'blocks', 'align',
 'bidi', 'paragraph']},
        {name: 'styles', groups: ['styles']},
        {name: 'colors', groups: ['colors']},
        {name: 'lite', groups: ['lite']},
        {name: 'about', groups: ['about']}
        ];
/**
* config.readOnly = true
* config.readOnly = false
* uncomment, based upon some conditions which may change according to the
 extension point configuration
*/
```

Figure 2: RSuite Edit defaults changed from CKEditor

Appendix
B

RSuite Edit built-in transforms

Topics:

This appendix describes the built-in transforms delivered by the RSuite Edit plug-in.

RSuite Edit default transform mappings

RSuite Edit includes transforms for:

- mapping DITA tags in RSuite files to XHTML tags for editing in RSuite Edit
- mapping XHTML mark-up in edited files back to DITA for saving in RSuite

In most cases, the original DITA mark-up is restored in the saved file. In two cases, the original DITA mark-up is replaced by a different but functionally equivalent mark-up. After the first editing session, the mark-up is stable.

The built-in transforms can be overridden by custom transforms. See *Configuring custom transforms* on page 14.

Transform mappings

Original DITA and DITA saved from XHTML	**DITA is mapped to this XHTML**
<b class="+ topic/ph hi-d/b ">	
<body class="- topic/body ">	
<bodydiv class="- topic/bodydiv ">	
<em class="+ topic/ph hi-d/i “ *Saved back to DITA as* <i class="+ topic/ph hi-d/i “>.	<em data-dita-class="+ topic/ph hi-d/i “>
<fn class="- topic/fn " >	^{ <a href="*link-to-cite-id*"/> } <cite id=”*cite-id*”>*footnote text*</cite>
<i class="+ topic/ph hi-d/i ">	<em data-dita-class="+ topic/ph hi-d/i “ >
<image class="- topic/image ">	<img data-dita-class="- topic/image ">
<li class="- topic/li ">	<li data-dita-class="- topic/p ">
<linethrough class="+ topic/ph d4p-formattingd/linethrough ">	<s id ="*id*">
<ol class="- topic/ol ">	<ol data-dita-class="- topic/ol ">
<p class="- topic/p ">	<p data-dita-class="- topic/p ">
<ph class="- topic/ph ">	
<ph class="- topic/ph " outputclass="ice-del ice-cts-1 “>	<del class="ice-cts-1 ice-del ">
<ph class="- topic/ph " outputclass="ice-ins ice-cts-1 ”>	<ins class="ice-cts-1 ice-ins ">
<section class= "- topic/section " >	
<sectiondiv class="- topic/sectiondiv">	
 Saved back to DITA as <b class="+ topic/ph hi-d/b ">.	
<subsection	<div

Original DITA and DITA saved from XHTML	**DITA is mapped to this XHTML**
class="- topic/topic subsection/subsection ">	data-dita-class="- topic/topic subsection/subsection ">
<title class="- topic/title ">	<h1 data-dita-class="- topic/title " >
<topic class="- topic/topic ">	
<ul class="- topic/ul ">	<ul data-dita-class="- topic/ul ">
<xref class="- topic/xref ">	<a data-dita-class="- topic/xref ">