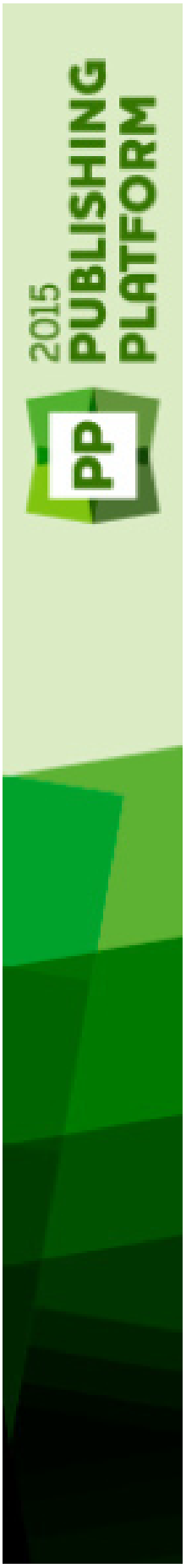


Quark Author Web Edition 2015 - System Administration Guide (v14.3)

2019/12/13

Contents

Introducing Quark Author configuration

Smart Content Schema

The Smart Content Schema version is 4.0. Any custom publishing process for Smart Content must point to this schema version. We suggest managing your own revisions to the schema as X.x.n, where X.x are major and minor Quark versions and n would serve as a placeholder for enumerating your implementation scheme.

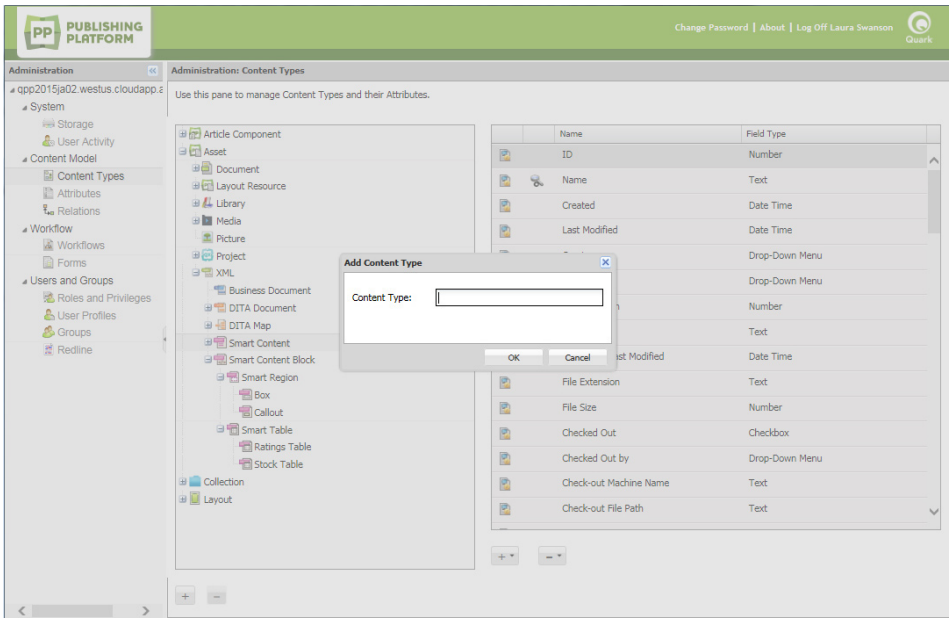
New Smart Content type creation configuration

Content type

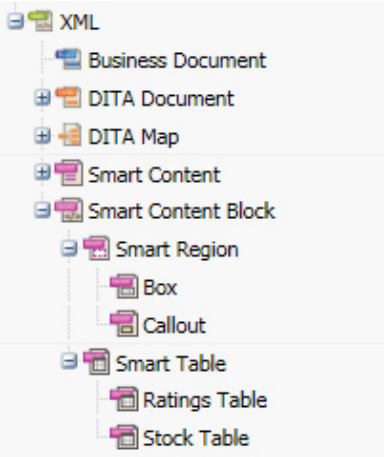
Creation

Create a new content type using the Platform Web Administrator.

To enable editing using Quark Author Web Edition, any new content type should be in a Platform content type hierarchy under “Smart Content”.



New component content types should be placed under “Smart Content Block”, which includes Smart Table and Smart Region.



Auto-detection

To configure the Platform Server to automatically detect the content type of the document being checked in from a local system to the Platform server and to automatically index new XML content type:

1. Declare a new custom mime type corresponding to the XML content type by adding an entry in the `{server}\ext\custom-xml-types-ext.xml` file:

```
<mime-type type="application/xml; format=outbreak"
  sub-class-of="application/xml; format=smartcontent">
    <xpath>/*[local-
name()='section']/@type='OutbreakNotice'</xpath>
</mime-type>
```

2. Update the `{server}\ext\content-mimetype-mappings-ext.xml` file to map the content type with the custom mime type defined in step 1:

```
<content-type name="Outbreak Notice">
  <mime-type>application/xml; format=outbreak</mime-type>
</content-type>
```

Publishing

The following sections describe how to configure Publishing Channels for documents of the new content type.

1. To create new publishing channels to be used for multi-channel preview, indexing and publishing of a new content type, update the `{server}\ext\ChannelConfig-ext.xml` file:

```
<channel id="OutbreakNoticeJpeg"
name="OutbreakNotice.JPEG.ZIP"
publishingProcess="smartDocToQxpRenderer" type="publish">
  <param name="ASSET_ID" ask="true"/>
  <param name="XSL_URI" ask="true"> classpath:OutbreakNotice
2QXPS.xslt</param>
  <param name="QXP_TEMPLATE_URI" ask="true">
classpath:OutbreakNoticeTe mplate.qxp</param>
  <param name="RENDER_FORMAT">JPEG</param>
  <param name="ANNOTATE_ERRORS">true</param>
  <param name="APPEND_ERRORS">true</param>
</channel>
<channel id="OutbreakNoticePdf" name="OutbreakNotice.PDF"
publishingProcess="smartDocToQxpRenderer" type="publish">
  <param name="ASSET_ID" ask="true"/>
  <param name="XSL_URI" ask="true">
```

```
classpath:OutbreakNotice2QXPS.xslt</param>
  <param name="QXP_TEMPLATE_URI" ask="true">
classpath:OutbreakNoticeTemplate.qxp</param>
  <param name="RENDER_FORMAT">PDF</param>
  <param name="ANNOTATE_ERRORS">true</param>
  <param name="APPEND_ERRORS">true</param>
  <param name="OUTPUT_STYLE" ask="true">Default PDF Output
Style</param>
</channel>
<channel id="OutbreakNoticeHtml"
name="OutbreakNotice.HTML.ZIP"
publishingProcess="smartDocToHtmlRenderer" type="publish">
  <param name="ASSET_ID" ask="true"/>
  <param name="XSL_URI" ask="true">
classpath:OutbreakNoticeToHTML.xslt</param>
</channel>
```

2. To map the publishing channels with a new content type, update the `{server}\ext\PublishingConfig-ext.xml` file:

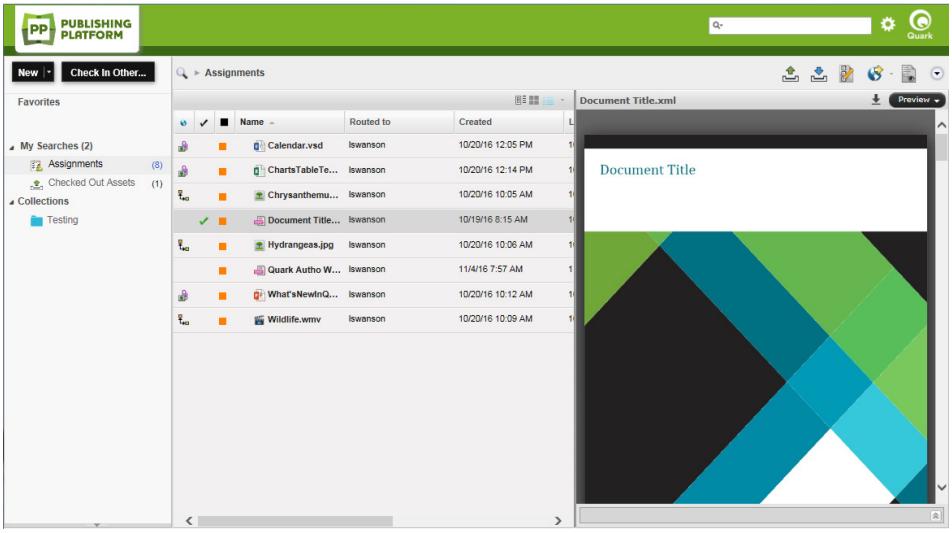
```
<content-type name="Outbreak Notice" applyToChildContentTypes
="true">
  <channelId>OutbreakNoticeJpeg</channelId>
  <channelId>OutbreakNoticePdf</channelId>
  <channelId>OutbreakNoticeHTML</channelId>
</content-type>
```

Indexing

To enable server side indexing for preview and thumbnail generation for documents of the new content type:

- Add mapping of new content type with publishing channel to be used for indexing by editing the `{server}\ext\indexingchannel-ext.xml` file:

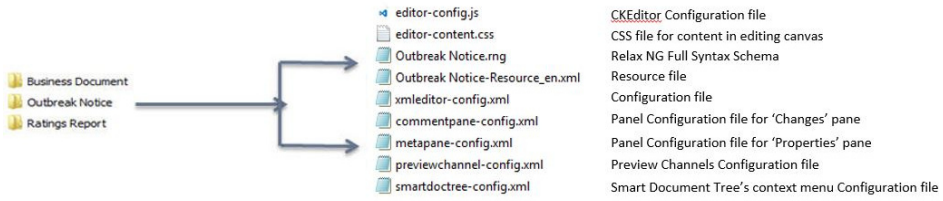
```
<contenttype-channel-mappings>
  <mapping contenttype="Outbreak Notice"
channel="OutbreakNoticeJpeg"/>
</contenttype-channel-mappings>
```



Document structure

Configuration

Quark Author Web Edition editor configuration for a content type is loaded from a folder with the same name as that of the content type. This folder is located at: `{server}\webapps\workspace\editor\config`



- `{content type}.rng`: `RelaxNG` files describing specialization of smart content schema for a document of a content type. Used to validate generated XML and includes Section Types, Section Hierarchy, Para and other block types and Metadata. Different `RelaxNG` files can be created for Tags, Meta and Structure definition.
- `{content type}-resource_xx.xml`: Files describing the language specific values of section and style names. Contains localized section names, block level and tag type names. Used for the placeholder text for section titles and also for the default (initial) text for the title and the body of sections.
- `{editor-content}.css`: A CSS file defining styles applied on content during editing. Defines section specific styles in editing canvas and block and tag type specific canvas styling.
- `{editor-config}.js`: A JavaScript file that overrides CKEditor default configurations like extra plugins, toolbar buttons etc. This is an optional file. If not present then the default configuration file is used.
- `{xmleditor-config}.xml`: Quark Author Web Edition editor configuration. Used to enable and position the **Action Panes**, the Header and the buttons in the editor, set the Preview, Componentization and Plugins configuration, tool-

bar configuration and editor settings. This is an optional file. If not present, then the default configuration file is used.

RelaxNG

The content type specific `RelaxNG` file defines the structure of a Smart Content document. This file also specifies different tags, block types, table types, list types and metadata that can be used in sections in the document.

Localized resources

The `Outbreak Notice-Resources_en.xml` file defines language specific values for Section and Style Names. Defines localized Section names, block level and Tag type names. Defines placeholder text for Section titles and the default (initial) text for Title and body of Sections.

```
<section type="OutbreakNotice" label="Outbreak Notice">
  <title placeholder="[Document Title Placeholder]"
default="Outbreak Notice Title"/>
  <body default="Document Content"/>
</section>
<section type="Synopsis" label="Synopsis">
  <title placeholder="[Synopsis Title Placeholder]"
default="Synopsis"/>
  <body default="Synopsis Content"/>
</section>
<section type="Assessment" label="Assessment">
  <title placeholder="[Assessment Title Placeholder]"
default="Assessment"/>
  <body default="Assessment Content"/>
</section>
<section type="Assessment Item" label="Assessment Item">
  <title placeholder="[Assessment Item Title]"
default="Assessment Item"/>
  <body default="Assessment Item Content"/>
</section>
<section type="Assessment Summary" label="Assessment Summary">
  <title placeholder="[Assessment Summary]" default="Assessment
Summary"/>
  <body default="Assessment Summary Content"/>
</section>
```

Editing canvas style configuration

The `editor-content.css:` File is a Cascading Style Sheet that defines CSS styles for the section title, body, table, ordered list, unordered list, para and Inline styles for a content-type.

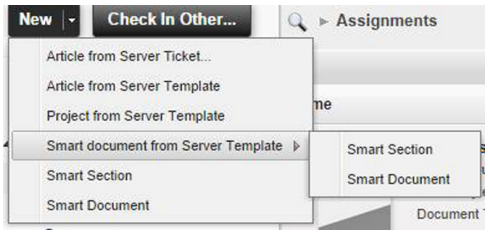
```
.Synopsis .body , .Assessment .body , . Assessment_Item , .
Assessment_Summary
{padding-left: 18px ;}
.title {
   font-size: 26.0pt ;
   font-family: " Calibri","sans -serif" ;
   color: #4f81bd ;
   text-transform: uppercase ;
   padding-bottom: 4px ;}
.Assessment .subtitle , .Synopsis .subtitle , .Recommendations
.subtitle {
   background: #4f81bd ;
   color: white ;
   padding-top: 3px ;
   padding-bottom: 3px ;}
.Assessment_Summary .subtitle {
   font-size: 11.0pt ;
   font-weight: bold ;
   font-style: italic ;
   color: #333 ;
   background: #dbe5f1 ;}
```

Workspace document instantiation

Use the `workspace-config.xml` file found here: `{server}\webapps\workspace\WEB-INF\classes\` to configure content types for document creation via the Platform Workspace using the `DocumentCreationSettings`.

- `id`: Specifies a unique id for the setting.
- `contentType`: Specifies the name of the platform specific content type of the created asset.
- `enableNew`: Enables the creation of an asset using the **New** menu in the workspace. The default value is `true`.
- `enableNewFromTemplate`: Enables the creation of an asset from a template. The default value is `true`.
- `assetBrowserId`: Specifies the unique ID indicating a particular `AssetBrowserSetting` to be used for filtering assets in the **New Document from Server Template** dialog. See the *Document instantiation from browser template configuration* section for a usage example.
- `starterTemplate`: Set this attribute to enforce the creation of a new asset by automatically using an existing asset as a template. The value is a URI to a platform asset to be used as the template. The value should be specified in the form `qpp://assetsbypath/[AssetPath]` or `qpp://assets/[AssetId]`

```
<DocumentCreationSettings>
  <DocumentCreationSetting id="new_qcd_menu_item"
contentType="CopyDesk Article"
          assetBrowserId ="QUARKCOPYDESK_BROWSER" type="QCD"/>
  <DocumentCreationSetting id="new_qxp_menu_item"
contentType="QuarkXPress Project"
          assetBrowserId ="QUARKXPRESSPROJECT_BROWSER"
type="QXP"/>
  <DocumentCreationSetting id="new_smart_doc_menu_item"
contentType="Smart Document"
          assetBrowserId ="SMARTDOC_BROWSER" type="XML"/>
  <DocumentCreationSetting id="new_smart_section_menu_item"
contentType="Smart Section"
          assetBrowserId ="SMARTSECTION_BROWSER" type="XML "/>
  <DocumentCreationSetting id="new_outbreak_menu_item"
contentType="Outbreak Notice"
          assetBrowserId ="OUTBREAK_BROWSER " type =" XML"
enableNew ="true"/>
</DocumentCreationSettings>
```



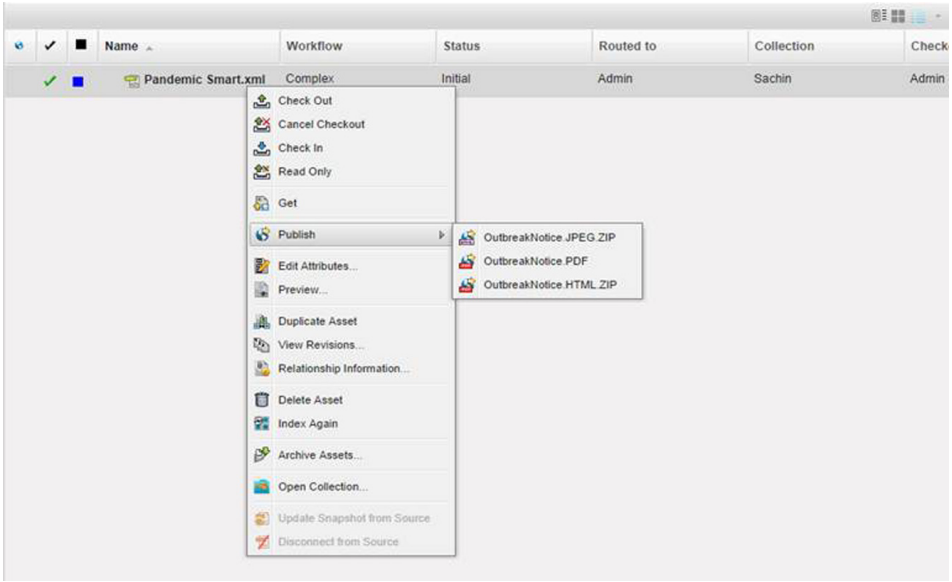
Workspace

Publishing

To enable publishing channels in the workspace, in the `{server}\webapps\workspace\WEB-INF\classes\Workspace-Config.xml` file:

- Add names of the new publishing channels to the value of the `enabledPublishingChannels` key.

```
<Add key="enabledPublishingChannels"

value="qxpPdf;qxpEpub;qxpAppStudio;qxpAppStudioPackage;busDocPdf;busDocHtml;busDocQxp;

busDocAppStudio;busDocAppStudioPackage;ditaDocTransformation;smartDocPdf;smartDocHtml;

smartTableHtml;smartTablePdf;OutbreakNoticeJpeg;OutbreakNoticePdf;OutbreakNoticeHtml"/>
```

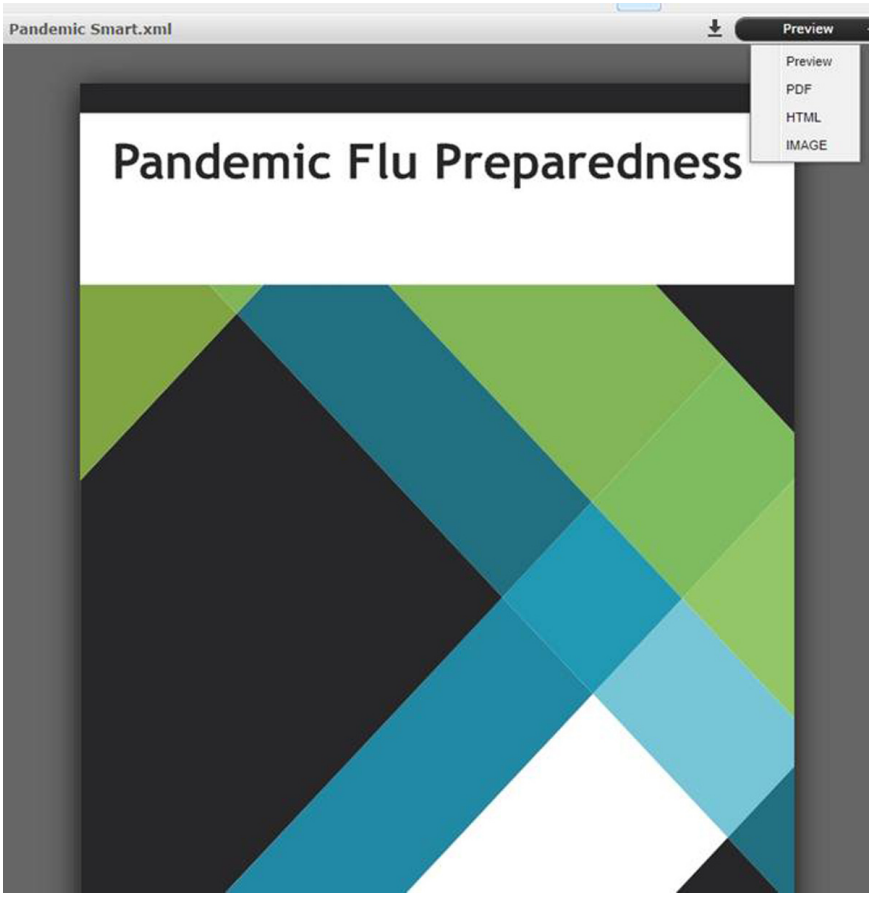


Multi-channel preview

To enable multi-channel preview for the “Outbreak Notice” content type in the `{server}\webapps\workspace\WEB-INF\classes\Workspace-Config.xml` file:

- Add new channel mapping by adding the following content:

```
<ChannelMapping contentType="Business Document"
applyToChildContentTypes="true">
  <Channels>
    <Channel id="OutbreakNoticePdf" outputFormat="PDF_ARCHIVE"
displayName="PDF"/>
    <Channel id="busDocHtml" outputFormat="HTML_ARCHIVE"
displayName="HTML"/>
    <Channel id="OutbreakNoticePdf"
outputFormat="IMAGE_ARCHIVE" displayName="IMAGE"/>
  </Channels>
</ChannelMapping>
```



Advanced configuration for document instantiation

Configuration

The Smart Content document level settings can be controlled using the `Workspace-Config.xml` file found in the application directory: `<Platform Server Home>\webapps\workspace\WEB-INF\classes`

Workspace

New menu configuration

Configure content types for document creation via the Platform Workspace using the `DocumentCreationSettings` element.

```
<DocumentCreationSettings>
  <DocumentCreationSetting id="new_qcd_menu_item"
contentType="CopyDesk Article"
          assetBrowserId ="QUARKCOPYDESK_BROWSER" type="QCD"/>
  <DocumentCreationSetting id="new_qxp_menu_item"
contentType="QuarkXPress Project"
          assetBrowserId ="QUARKXPRESSPROJECT_BROWSER"
type="QXP"/>
  <DocumentCreationSetting id="new_smart_doc_menu_item"
contentType="Smart Document"
          assetBrowserId ="SMARTDOC_BROWSER" type="XML"/>
  <DocumentCreationSetting id="new_smart_section_menu_item"
contentType="Smart Section"
          assetBrowserId ="SMARTSECTION_BROWSER" type="XML "/>
</DocumentCreationSettings>
```

- `Id`: Specifies the unique id of the setting.
- `contentType`: Specifies the name of the platform specific content type of the created asset.
- `AssetBrowserId`: Specifies the id of the asset browser settings to be used for configuring and filtering the asset picker dialog for selecting the template.
- `starterTemplate`: Use this attribute to instantiate a new asset using an existing asset. The value of this attribute is the URI to the platform asset. The value could be specified in one of the following formats:
  - `qpp://assetsbypath/[AssetPath]`

- `qpp://assets/[AssetId]`

- `enableNew`: Use to enable the creation of an asset of the specified content type using the **New** menu. The default value is `true`.

```
DocumentCreationSetting id="new_smart_doc_menu_item"
    contentType="Smart Document"
    assetBrowserId="SMARTDOC_BROWSER"
    type="XML"
    enableNew="true"/>
```

Set its value to false to hide the 'Smart Document' item from the **New** menu.

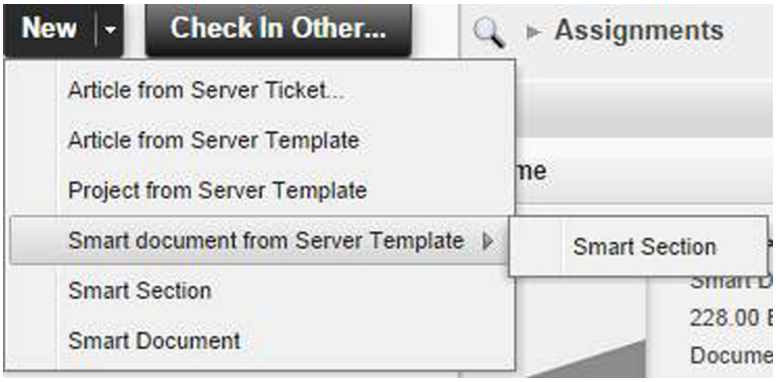
```
DocumentCreationSetting id="new_smart_doc_menu_item"
    contentType="Smart Document"
    assetBrowserId="SMARTDOC_BROWSER"
    type="XML"
    enableNew="false"/>
```

- `enableNewFromTemplate`: Use to enable the creation of an asset from an existing asset. The default value is true. The following example shows the **Smart Document** item in the **New > Smart document from Server Template** menu.

```
DocumentCreationSetting id="new_smart_doc_menu_item"
    contentType="Smart Document"
    assetBrowserId="SMARTDOC_BROWSER"
    type="XML"
    enableNewFromTemplate="true"/>
```

Set its value to false to hide the 'Smart Document' item from the **New > Smart document from Server Template** menu.

```
DocumentCreationSetting id="new_smart_doc_menu_item"
    contentType="Smart Document"
    assetBrowserId="SMARTDOC_BROWSER"
    type="XML"
    enableNewFromTemplate="false"/>
```



Document template configuration

Documents of new content type can use the *config* files of an existing document by specifying the mapping in the `XmlEditorConfigMappings` element. This allows you to specify the folder mapping between Platform content type and Smart Content configuration files.

This is an optional element. If it is not specified, the exact name of the content type is used as the folder name for searching configuration files.

The 'Smart Document Template' content type maps with the same configuration files that are used for 'Smart Document' which means when a new document of type 'Smart Document Template' is created then the same configuration files that are being used for 'Smart Document' content type will be used for this also.

- `contentType`: Specifies the Platform content type that would be mapped.
- `xmlEditorConfig`: Specifies the folder name that contains the set of configuration files and the map with the Platform content type.

```
<XmlEditorConfigMappings>
  <XmlEditorConfigMapping contentType="Smart Document Template"
    xmlEditorConfig="Smart Document"/>
</XmlEditorConfigMappings>
```

Document instantiation from pre-defined template configuration

If the template file, `ReportingTemplate.xml`, is checked-in to the server and you need to configure the starter template for content type 'Smart Document' so that when a new document of type 'Smart Document' is created, it has pre-populated sections the same as in the template:

1. Update the `DocumentCreationSettings` key for content type 'Smart Document' to specify a starter template URI.

```
<DocumentCreationSettings id="new_smart_doc_menu_item"
  contentType="Smart Document"
  assetBrowserId="SMARTDOC_BROWSER"
  type="XML"

starterTemplate="qpp://assetsbypath/Home/ReportingTemplate.xml
"/>
```

2. Create a new Smart Document via the **New** menu. A new instantiated document will have the same sections as in the starter template

Document instantiation from browser template configuration

This allows users to create new documents based on existing templates in the Platform Server. When the user chooses this option, the asset picker dialog is shown with the server templates configured and the user has access to them.

If a new content type 'Smart Document Template' is created and the **New** menu configured in such a way that when the **Smart Document > Smart Document from server template** menu option is selected, documents belonging to the "Smart Document Template" content type should be shown in an asset browser dialog.

To accomplish this:

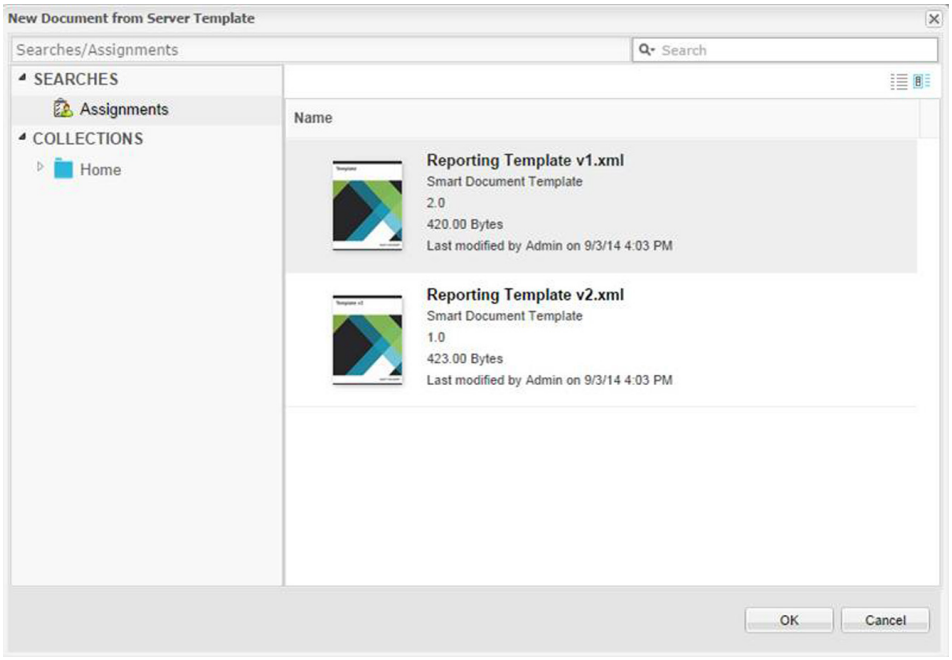
1. Create a new asset browser setting that searches for content type "Smart Document Template":

```
<AssetBrowserSetting id="SMARTDOCTEMPLATE_BROWSER"
searchForContentType="Smart Document Template"/>
```

2. Update the document creation setting for content type "Smart Document" to use the `SMARTDOCTEMPLATE_BROWSER` browser id for searching the template.

```
<DocumentCreationSetting id="new_smart_doc_menu_item"
  contentType="Smart Document"
  assetBrowserId="SMARTDOCTEMPLATE_BROWSER"
  type="XML"/>
```

3. Create a smart document using the server template (**New > Smart document from Server Template > Smart Document**). The asset browser dialog will show only documents of type "Smart Document template".



Asset browser settings

Specify the settings of the asset picker dialog using the `AssetBrowserSettings` element. These settings can be used to invoke the asset picker dialog.

- `id`: Specifies a unique id for the setting.
- `searchForContentType`: Specifies the platform specific content type name to identify assets of specified type. List of content types can be specified by comma separated values.
- `enableCollectionBrowser`: Use to enable collection browsing in the asset picker dialog. The default value is `true`.
- `enableSavedSearches`: Use to enable saved searches in the asset picker dialog. The default value is `true`.
- `enableQuickSearch`: Use to enable quick search in the asset picker dialog. The default value is `true`.
- `includeChildContentTypes`: Use to allow a search to include assets of child content types. The default value is `false`.
- `filter`: Specifies the additional filters to refine the results. (For example: file extension must be JPEG or assets that have a workflow status of `Published`.) You can specify multiple filters using ";" as a separator:

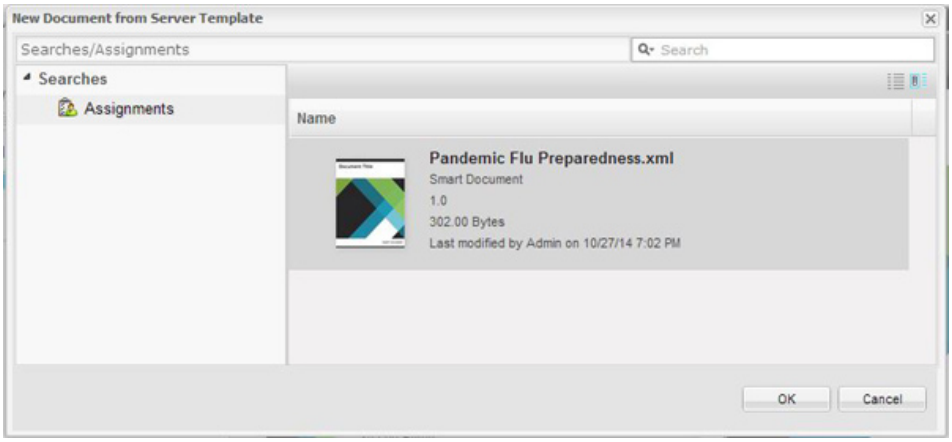
  ```
  filter="Is checked out=true; Routed to=Admin"
  ```

  You can specify multiple values for domain type attribute using "," as a separator:

  ```
  filter="Routed to=User1,User2,User3"
  ```

```
<AssetBrowserSettings>
  <AssetBrowserSetting id="QUARKCOPYDESK_BROWSER"
searchForContentType="QuarkCopyDesk Article Template"/>
  <AssetBrowserSetting id="QUARKXPRESSPROJECT_BROWSER"
searchForContentType="QuarkXPress Project Template"/>
  <AssetBrowserSetting id="PICTURE_BROWSER"
```

```
searchForContentType="Picture" includeChildContentTypes="true"/>
  <AssetBrowserSetting id="DATATABLE_BROWSER"
searchForContentType="Smart Table"/>
  <AssetBrowserSetting id="EXCEL_BROWSER"
searchForContentType="Microsoft Excel,Microsoft Excel Template"/>
  <AssetBrowserSetting id="SMARTDOC_BROWSER"
searchForContentType="Smart Document"
enableCollectionBrowser="false"/>
  <AssetBrowserSetting id="REGION_BROWSER"
searchForContentType="Smart Region"/>
  <AssetBrowserSetting id="EXCEL_TABLE_BROWSER"
searchForContentType="Microsoft Excel,Microsoft Excel
Template,Microsoft Excel Table"
includeChildContentTypes="false"/>
  <AssetBrowserSetting id="EXCEL_CHART_BROWSER"
searchForContentType="Microsoft Excel,Microsoft Excel
Template,Microsoft Excel Chart"
includeChildContentTypes="false"/>
</AssetBrowserSettings>
```



Document save configuration

Attribute mapping and revision settings

The `AttributeMapping` element for each content type present in the `workspace-config.xml` file allows you to:

- Specify the document attributes that should be mapped to Platform attributes.
- Specify save configuration when a document is saved to the Platform Server.

```
<AttributeMapping>
  <ComponentTypes>
    <ComponentType name="Smart Section">
      <RevisionSettings>
        <RevisionSetting operation="SaveDocumentRevision">
          <SaveSilently>false</SaveSilently>
          <SaveAsMinorVersion>false</SaveAsMinorVersion>
        </RevisionSetting>
        <RevisionSetting operation="SaveDocument">
          <SaveSilently>false</SaveSilently >
          <SaveAsMinorVersion>false</SaveAsMinorVersion>
        </RevisionSetting>
        <RevisionSetting operation="ExportComponent">
          ......
        </RevisionSetting>
        <RevisionSetting operation="SaveComponent">
          <SaveSilently>false</SaveSilently>
          <SaveAsMinorVersion>false</SaveAsMinorVersion>
        </RevisionSetting>
      </RevisionSettings>
      <Attributes>
        <Attribute name="Text preview"
xpath="/smart:section/smart:title"
indexingOption="ALL_VERSIONS"/>
        <Attribute name="Global ID" xpath="/smart:section/@id"
indexingOption="ALL_VERSIONS"/>
        <Attribute name="File extension" value="xml"
indexingOption="ALL_VERSIONS"/>
        ......
      </Attributes>
    </ComponentType>
  </ComponentTypes>
</AttributeMapping>
```

Revision settings

The `RevisionSettings` element specifies the document and component versioning scheme (major or minor) and configuration with regard to the **Save** dialog visibility while saving a document.

- `Operation`: supported values are:
  1. `SaveDocument`: Saving a document to the server. Settings value referred to when a user clicks the **Save and Close** button in the Editor.
  2. `SaveDocumentRevision`: Saving a document revision to the server. Settings value referred to when a user clicks the **Save** button in the Editor.
  3. `ExportComponent`: Exporting a section of this component type from document of another component type and saving it as a document to the server. Settings value referred to when a user executes a **Create Component** operation in the Editor.
  4. `SaveComponent`: Saving a component as a new document version to the server. Settings value referred to when a component document is checked out inline in the main document and then checked in.
- `SaveSilently`: Specifies whether the document or component is saved without showing the **Save** dialog or not.
- `SaveAsMinorVersion`: Specifies whether the document or component is saved as a minor version or not.

Attribute mapping

The `Attributes` element specifies the mapping of data from the document to the Platform Server attributes. The values of the Platform attributes can be specified either as static text or an XPath to the content in a document.

- `name`: Specifies the name of the Platform Server attribute.
- `value`: Specifies the static attribute value.
- `xpath`: Specifies the XPath to be used for setting the Platform attribute value.

➡ You can either set XPath or a static value.

- `indexingOption`: Supported values are:
  1. `INITIAL_VERSION`: Set this value to trigger indexing only for the first version of the document. This is the default value of the `indexingOption`.
  2. `ALL_VERSIONS`: Set this value to trigger indexing for every revision
- `inheritValueFromTemplate`: Specifies whether the attribute value is inherited from the server template. The default value is `false`.

Attribute mapping configuration

If you want to configure some attributes so that attribute values are fetched from the document for the “Smart Document” content type:

- Update the `Attributes` key for content type ‘Smart Document’:

```
<Attributes>
  <Attribute name="Text preview"
xpath="/smart:section/smart:title"
indexingOption="ALL_VERSIONS"/>
  <Attribute name="Global ID" xpath="/smart:section/@id"
indexingOption="ALL_VERSIONS"/>
  <Attribute name="Title" xpath="/smart:section/smart:title"
```

```
indexingOption="ALL_VERSIONS"/>
  <Attribute name="File extension" value="xml"
indexingOption="ALL_VERSIONS"/>
  <Attribute name="TextAttr"

xpath="/smart:section/smart:body//smart:meta/smart:attribute[@
name='disease']/smart:value"
     indexingOption="ALL_VERSIONS"/>
  <Attribute name="DateAttr" value="2014-08-14"
indexingOption="ALL_VERSIONS"/>
  <Attribute name="NumberAttr" value="15"
indexingOption="ALL_VERSIONS"/>
  <Attribute name="BooleanAttr"

xpath="/smart:section/smart:body//smart:meta/smart:attribute[@
name='transmission']/smart:value"
     indexingOption ="ALL_VERSIONS"/>
  <Attribute name="DateTimeAttr"

xpath="/smart:section/smart:body//smart:meta/smart:attribute[@
name='host']/smart:value"
     indexingOption ="ALL_VERSIONS"/>
</Attributes>
```

When a document of type “Smart Document” is checked in, the value of attributes like `Text preview` and `Title` are fetched from the document based on the XPath and get saved along with the document.

Auto save configuration

The time interval for auto saving a document is configured in the `workspace-config.xml` file found in the application directory: `{QPP Server}\webapps\workspace\WEB-INF\classes`.

- To periodically save a draft copy of the document during an editing session: `<Add key="enableAutoSave" value="true"/>`
- To configure how often changes are saved as a draft, specify the time interval in milliseconds for the auto save: `<Add key="autoSaveInterval" value="30000"/>`

➡ This should not be less than the XML Editor defined minimum value of 5 seconds.

- To define whether to create an auto save revision on server or on local storage. Possible values are server or local. `<Add key="autoSaveRevisionStorage" value="server"/>`
- To define snapshot limit for an individual asset—> `<Add key="snapshotLimit" value="50"/>`

Offline editing configuration

The offline editing feature allows a user to edit checked-out documents when not connected to the Platform server.

➡ This feature is only supported in the Chrome browser and requires HTTPS configuration.

To disable offline editing for all users:

1. Open the `Workspace-config.xml` file located at `{server}\webapps\workspace\WEB-INF\classes`.
2. Set the value for `enableOfflineMode` to **false**.

To update the application cache for all users:

1. Open the `Deployment.properties` file located at `{server}\ext`.
2. Increment the value for the deployment.version.

Grammar and spell checking configuration

Entries for the configuration of grammar and spell checking can be found at the following two locations:

- The `xmleditor-config.xml` file located in the `{server}\webapps\workspace\editor\config\` folder.

  A new entry to declare the `ProofReader` object has been added. When creating new document types, your specialized `xmleditor-config.xml` files must also contain this entry if you want to take advantage of the grammar and spell checking features of the LanguageTool.

  ```
  <item>ProofReader</item>
  ```

- The `editor-config.xml` file located in the `server}\webapps\workspace\editor\js\thirdparty\ckeditor\` folder.

  Use this file to configure the behavior of the LanguageTool features.

Disabling

Disabling all proofing services on startup

The LanguageTool spell-checking feature is enabled by default when a document is loaded in Quark Author.

To disable the default behavior and force users to manually activate the proofing services:

1. Open the `editor-config.js` file located in the `{server}\webapps\workspace\editor\js\thirdparty\ckeditor` folder.
2. Locate the option by searching for the string “proofreader_autoStartup”. Set to false to disable the spell checker on startup.

   ```
   config.proofreader_autostartup = false;
   ```

Disabling grammar check

The LanguageTool grammar feature is enabled by default when a document is loaded in Quark Author.

To disable this feature for all users:

1. Open the `proofreader-servlet.xml` file located in the `{server}\webapps\proofreader\WEB-INF` folder.
2. Set the following property to false:

```
<property name="enableGrammarCheck" value="false"></property>
```

Disabling specific grammar rules

Quark Author provides a configuration option to disable specific grammar rules.

To define a list of rules that should be disabled:

1. Open the `proofreader-servlet.xml` file located in the `{server}\webapps\proofreader\WEB-INF` folder.
2. Set the following property:

```
<property name="disabledRules">
   <list>
      <value>EN_QUOTES</value>
   </list>
</property>
```

➡ The names of the rules are available in the `grammar.xml` file for each language located in the `{server}\webapps\proofreader\WEB-INF\classes\org\languagetool\rules` folder.

Configure grammar and spell checking capabilities

There are various configurations that can be applied to control the grammar and spell-checking capabilities in Quark Author.

Dictionaries

Adding dictionaries for languages supported by LanguageTool

The grammar and spell-checking features in Quark Author are enabled using a third-party library, LanguageTool (*http://languagetool.org/*). The library supports spell checking for many languages. Some languages are excluded from the Quark Publishing Platform distribution due to licensing restrictions. It's easy to add support for these languages by copying the appropriate dictionary files from the LanguageTool distribution into your Platform installation.

To add a dictionary for a specific language:

1. Download and extract the latest zip file of the LanguageTool snapshot build from the following location:

   https://languagetool.org/download/snapshots/?C=M;O=D

2. Copy the language specific folder from `<LanguageToolSnapshot>\org\languagetool\resource`. For example, the folder with the name 'de' contains dictionary files corresponding to German.
3. Paste the folder into the Platform installation tree:

   <Platform-Server>/webapps/proofreader/WEB-INF/classes/org/languagetool/resource

4. Add an entry for the new language in the `language-module.properties` files located in the `<Server>/webapps/proofreader/WEB-INF/classes/META-INF/org/languagetool` folder.

➡ The entry should contain the fully qualified name of the language-specific class for the new language. The language-specific class files are available in the *`<Server>/webapps/proofreader/WEB-INF/classes/org/languagetool/language`* folder.

➡ For example, the name of the class for German is "GermanyGerman" and its fully qualified name is "org.languagetool.language.GermanyGerman", so the entry to be added will be `languageClasses=org.languagetool.language.GermanyGerman.`

5. Repeat the process for all Platform Server nodes when using a multi-server configuration.

6. Restart Platform Server. The new language will now be available for grammar and spell checking.

Adding a system-wide corporate dictionary

A corporate dictionary containing words specific to an entire organization can be added by extending the default dictionary for a specific language. As a best practice, Quark recommends capturing any specific terms ahead of the spell checker deployment.

Extend the dictionary by adding new words to the `spelling.txt` file available in the *`<Server>/webapps/proofreader/WEB-INF/classes/org/languagetool/resource/en/hunspell`* folder.

➡ Here, the "en" folder corresponds to the English language.

A `spelling.txt` file exists for every language and new words should be added for each supported language in an installation.

➡ The words in `spelling.txt` are ignored during spell checking and are available as suggestions for invalid words.

Viewing and updating words in a User Dictionary

Currently, there is no user interface to view and edit words in a user's dictionary. Viewing and editing terms in a user dictionary is achieved by making a REST request to Platform Server.

Invoke the following URL to view dictionary words for an Admin user:

```
http://<Server>:61400/rest/service/admin/preferencevalues?preferences=Proof Reader User Dictionary&loginname=Admin&loginpassword=Admin
```

Invoke the following request to update the dictionary words for a user:

```
http://<Server>:61400/rest/service/admin/preferencevalues?op=update&preferencevaluelist=<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <preferenceValueList> <preferenceValue preferenceName="Proof Reader User Dictionary">Errer,Woord</preferenceValue></preferenceValueList>&loginname=Admin&loginpassword=Admin
```

➡ The list of words to be updated should be in comma separated format.

Ignoring words

Ignoring words globally

Add the list of words to be ignored during spell check for all users to the `ignore.txt` file available for each language in the `<Server>/webapps/proofreader/WEB-INF/classes/org/languagetool/resource/en/hunspell`folder.

➡ The words specified in the `ignore.txt` file are ignored during spell checking but are not available as suggestions for invalid words.

Ignoring words in camel case

The default installation marks invalid words in Camel Case. This behavior can be customized to ignore invalid words which are in Camel Case.

To ignore invalid words in Camel Case:

1. Open the `<en>_<US>.info` file located in the `{server}\webapps\proofreader\WEB-INF\classes\org\languagetool\resource\en\hunspell` folder.
2. Set the following configuration:

   `fsa.dict.speller.ignore-camel-case=true`

➡ There are separate folders and a `.info` file for each language.

Ignoring words in upper case

The default installation marks invalid words in Upper Case. This behavior can be customized to ignore invalid words which are in Upper Case.

To ignore invalid words in Upper Case:

1. Open the `<en>_<US>.info` file located in the `{server}\webapps\proofreader\WEB-INF\classes\org\languagetool\resource\en\hunspell` folder.
2. Set the following configuration:

   `fsa.dict.speller.ignore-all-uppercase=true`

➡ There are separate folders and a `.info` file for each language.

Specifying correct words that should be flagged

A user may have a list of words that they want flagged as incorrect, even though the spell checker would normally accept them.

To specify a list of correct words the spell checker should flag, add the list of words to the `prohibit.txt` file available for each language in the `<Server>/webapps/proofreader/WEB-INF/classes/org/languagetool/resource/en/hunspell`folder.

Adding support for a new language

Though LanguageTool provides support for numerous languages, it's possible for anyone to add new languages.

The process for adding a new language is described at the LanguageTool wiki:

http://wiki.languagetool.org/adding-a-new-language

Tokenization configuration

Configuring the maximum number of suggestions

The maximum number of suggestions that are shown for an invalid word is seven.

To change this number:

1. Open the `editor-config.js` file located in the `{server}\webapps\workspace\editor\js\thirdparty\ckeditor` folder.
2. Locate the option by searching for the string "proofreader_maxSuggestionCount". Set the value to the number of desired suggestions shown for an invalid word.

   ```
   config.proofreader_maxSuggestionCount = 7;
   ```

Configuring the number of paragraphs

To specify the number of paragraphs that are processed and grouped in the event data:

1. Open the `editor-config.js` file located in the `{server}\webapps\workspace\editor\js\thirdparty\ckeditor` folder.
2. Locate the option by searching for the string "proofreader_chunkSize". Set the value to the number of desired paragraphs.

   ```
   config.proofreader_chunkSize = 100;
   ```

Configuring wait time

To specify the time in milliseconds between the last changes on the canvas and the processing for a data block:

1. Open the `editor-config.js` file located in the `{server}\webapps\workspace\editor\js\thirdparty\ckeditor` folder.
2. Locate the option by searching for the string "proofreader_waitTime". Set the value to the number of milliseconds to wait.

   ```
   config.proofreader_waitTime = 3000;
   ```

Specifying the default proofreading language

To configure the default proofreading language:

1. Open the `editor-config.js` file located at `{server}\webapps\workspace\editor\js\thirdparty\ckeditor.`
2. Locate the option by searching for the string **proofreader_defaultLanguage**.

3. Set the default proofreading language.

```
config.proofreader_defaultLanguage = "en-CA";
```

➡ The proofing language is set to “English(Canadian)” by default, when a document is opened or checked out.

Excluding block elements from spell checking

To configure a list of block type elements that is to be excluded from proofreading:

1. Open the `editor-config.js` file located at `{server}\webapps\workspace\editor\js\thirdparty\ckeditor.`
2. Locate the option by searching for the string **proofreader_ignoreElements**.
3. Specify the list of elements to ignore while proofreading.

```
config.proofreader_ignoreElements  = [
{
  element : 'region’,
  elementType : 'box’,
},
];
```

➡ All regions of type “box” will be excluded from proofreading (irrespective of its location in the document).

Excluding read-only text from spell checking

To prevent proofreading of text inside a read-only block:

1. Open the `editor-config.js` file located at `{server}\webapps\workspace\editor\js\thirdparty\ckeditor.`
2. Locate the option by searching for the string **proofreader_ignoreReadOnlyElements**.
3. Set the option to **true**.

```
config.proofreader_ignoreReadOnlyElements = true;
```

Enabling the Web Spell Checker plugin

Integration with the Spell Checker plugin allows the editor to spell check the document and enable the **Spell Check As You Type** feature in Quark Author Web Edition.

This technology is used by CKEditor by default and typically requires a separate license. Disabling the Web Spell Checker plugin avoids the ads that come with the free version and minimizes third party license costs required to remove them. When enabled, this technology securely sends information to the Amazon public cloud. For customers choosing to use this technology instead of LanguageTool, follow the steps below.

By default this plugin is disabled.

To enable this plugin on a Platform Server node:

1. Open the `editor-config.js` file located at `{server}\webapps\workspace\editor\js\thirdparty\ckeditor.`
2. Locate the disabled plugins list by searching for the string **removePlugins**.
3. Remove the entries for **scayt** and **wsc** from `config.removePlugins`.

```
CKEDITOR.editorConfig = function(config)
{
  config.enterMode = CKEDITOR.ENTER_DIV;
  config.shiftEnterMode = CKEDITOR.ENTER_DIV;
  config.fullPage = false;
  config.defaultLanguage = "en";
  config.dialog_buttonsOrder = "ltr";
  config.allowedContent = true;
  config.pasteFilter = null;
  config.removePlugins =
'elementspath,forms,magicline,image,div,showborders,tabletools
,tab,scayt,wsc';
}
```

4. Open the `xmleditor-config.xml` file located in the `{server}\webapps\workspace\editor\config\ folder.`
5. Add an entry for the **Scayt**:

```
<tab id="home" type="main"
  <items>
    <item>Undo</item>
    <item>Redo</item>
     ....
    <item>viewxml</item>
    <item>Scayt</item>
  </items>
</tab>
```

6. Save the changes.

➡ If the settings in the `editor-config.js` file have been customized for different document types, then repeat these changes for each document type. Document-type specific `editor-config.js` files are located at `{server}\webapps\workspace\editor\config\<Document type>.`

Date/time in comment tooltip configuration

The `showDateTimeInTooltip` element is used to show/hide the date and time value in the comment tool tip.

1. Open the `editor-config.js` file located in the `{server}\webapps\workspace\editor\js\thirdparty\ckeditor` folder.
2. Locate the option by searching for the string “showDateTimeInTooltip”. Set to false to hide the comment creation date and time information from the comment tooltip.

   ```
   config.commenting_showDateTimeInTooltip = false;
   ```

Adding domain values to Platform attributes

You can configure the users' ability to add values to single valued, non-hierarchical drop-down type attributes. Once configured, the user uses the **CHECK IN** dialog to add these values.

To configure the authors' ability to add new metadata values:

1. Open the `metapane-config.xml` file located in the `{server}\webapps\workspace\editor\config\` folder.
2. Set the value for `editable` to `true`.

```
<meta-configs>
  <meta-config name ="country" priority="10" sort="DESC" >
    <data>
      <providerref idref="CountryDomainProvider"
editable="true">
        <valueField>id</valueField>
        <displayText>text</displayText>
      </providerref>
    </data>
  </meta-config>
</meta-configs>
```

See the "Working with attributes" section in the *Platform User Guide* for more information.

Smart content editor configuration

Use the `xmleditor-config.xml` file located in the `{server}\webapps\workspace\editor\config\` folder to configure Smart Content Editor level configuration. Place this file under each content type folder to override the configuration settings based on content type.

The `xmleditor-config.xml` file is used to define default settings for all doctypes and can be embedded in the content type folder. Smart Document and Smart Section content types have their own `xmleditor-config.xml` files, as well as CSS, RNG and language-specific resource files (for example the English resource file for Smart Document is named `Smart Document-Resource_en.xml` and would be in the `{server}\webapps\workspace\editor\config\Smart Document` folder).

Among other Editor settings, you can configure this file to specify the following:

- Enable and position the action panes, the header and the buttons in the Editor
- Preview and componentize configurations
- Configure Plugins
- Configure toolbar
- Configure spell checking

Supports overriding of selective configuration elements which means if other elements are not specified then the value will be picked from the file in this location.

Panel configuration

The `Panels` element is used to specify the number of default panels to be viewed and the panels appearance and location when first opened.

- `collapseLeftPanel`: Specifies that when a document is opened the left side panels are collapsed.
- `hideLeftPanel`: Specifies that when a document is opened the left side panels are hidden.
- `collapseRightPanel`: Specifies that when a document is opened the right-hand panels are collapsed.
- `hideRightPanel`: Specifies that when a document is opened the right-hand panels are hidden.
- `config`: Optional attribute to specify the path of the panel configuration file.

- `position`: Specifies the location of the panel in the Editor. `Left` and `Right` are the supported values. If multiple panes are specified in a single location the panes are rendered based on the sequence specified here.
- `mode`: Specifies the document mode supported by the panel. Possible values are `any`, `edit` and `readOnly`.
- `view-name`: Specifies the name of the **extjs** based view associated with the panel. Possible values are `smartdoctree`, `xmlPreview`, `commentView`, `notePanel`, `metaPanel`, `indexPanel`, `refersToPanel`, `referencePanel`.
- `controller`: Specifies the name of the **extjs** based controller class managing panel interactions.

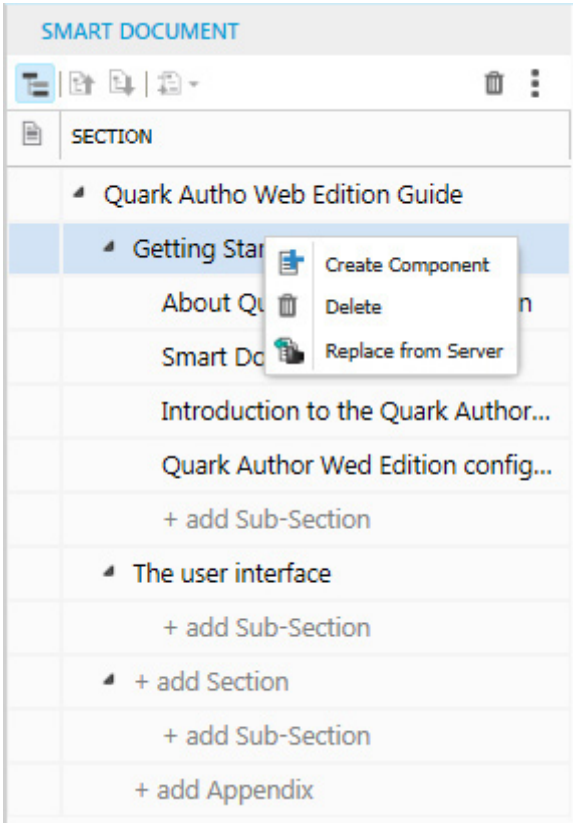
```
<panels collapseLeftPanel="false" hideLeftPanel="false">
  <panel position="left"
    mode="any"
    view-name="smartdoctree"

controller="com.quark.kestrel.feature.controller.SmartDocumentCon
troller"
    config="smartdoctree-config.xml"/>
  <panel position="right"
    view-name="xmlPreview"

controller="com.quark.kestrel.feature.controller.PreviewPaneContr
oller"
    config="previewchannel-config.xml"/>
</panels>
```

Componentization

The `bursting-config` element is used to specify the rules for componentization. Each bursting rule specifies the section type that can be componentized into a Platform specific content type.

- `section-type-path`: Specifies the section XPath which can be componentized as a separate asset.
- `content-type`: Specifies the corresponding Platform specific content type of the section.

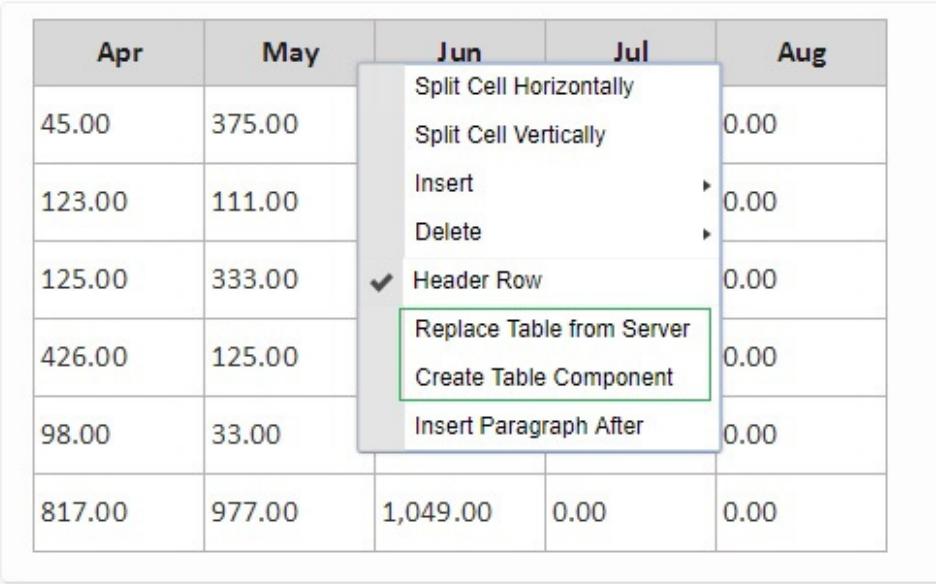
```
<bursting-config>
  <bursting-rule section-type-path="/document/section" content-
type="Smart Section"/>
</bursting-config>
```



The `block-bursting-config` element is used to specify the bursting rules for block-level components like regions and tables. If a bursting rule matches a region, table, or other content type, then the specified type can be exported or imported as a separate component, enabling reuse for that block type in any number of documents. UI commands such as “Edit component”, “Replace from Server” etc. are also enabled for any block-level component defined using this pattern.

- `selector`: Specifies the element type and section path where the component reference is allowed.
- `blockpicker-setting`: Specifies asset browser behavior for new content types.

```
<configuration>
  <blockpicker-settings>
    <blockpicker-setting contentType="Smart Region"
      assetBrowserId="REGION_BROWSER"/>
    <blockpicker-setting contentType="Smart Table"
      assetBrowserId="DATATAVKE_BROWSER"/>
  </blockpicker-settings>
  <block-bursting-config>
    <block-bursting-rule content-type="Smart Region">
      <selector element-name="region"/>
    </block-bursting-rule>
    <block-bursting-rule content-type="Smart Table">
      <selector element-name="table"/>
    </block-bursting-rule>
  </block-bursting-config>
</configuration>
```



Changes pane configuration

Configure the **Changes** pane to show or hide toolbar buttons in the panel by editing the `commentpane-config.xml` file located at `{server}\webapps\workspace\editor\config\`.

Place this file under each content type folder to override the configuration settings based on content type.

`The <toolbar>` element contains a list of buttons that appear in the panel's toolbar at the top. Default buttons that appear in the toolbar can also be removed by removing its corresponding entry.

```
<toolbar>
        <button id="add-comment-button"  iconCls="add-comment-
icon" />
        <button id="delete-comment-button" iconCls="delete-
comment-icon"/>
        <button id="comment-pane-refresh-button"
iconCls="refresh-icon" align = "right"/>
    </toolbar>
```

- `id`: Unique ID of the button. Do not change the ID for default buttons
- `icon:` URL for the icon image
- `iconCls:` CSS class for the icon image
- `align`: Aligns button in the toolbar. Optional (**left**,**right**).
- `separator`: If **true** then adds a separator after toolbar button. Optional.

`The <user-source>` element is used to specify a data provider that can be used to provide a list of authors to be displayed in the user filter. The provider should return a JSON object containing the following data to be displayed in the UI:

- `authorId`: Unique author name which will be persist in XML.
- `displayText`: the author's name to be displayed in the UI.

- `online`: indicates whether the user is online or not.

```
<user-source scope="CommentAuthorProvider"
callback="getAuthors"/>
```

Properties pane configuration

Configure the **Properties** pane by editing the `metapane-config.xml` file located at `{server}\webapps\workspace\editor\config\`.

Place this file under each content type folder to override the configuration settings based on content type.

Edit the file to set the available configuration settings:

- Hide/show toolbar buttons and their respective icons
- Align buttons right or left
- Modify display order for different metadata fields
- Set the sort order for values in a list
- Set fields to read-only or hidden by context
- Define a data provider to obtain managed values

Configuring toolbar

The `<toolbar>` element contains a list of `<button>` elements that appear in the panel's toolbar at the top.

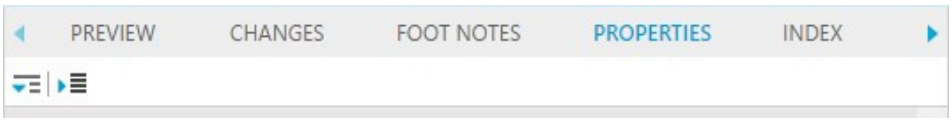
- `id`: Unique ID of the button.
- `icon`: URL for the icon image.
- `iconCls`: CSS class for the icon image.
- `align`: aligns button in toolbar. Optional (**left**,**right**).
- `separator`: adds a separator after the toolbar button. Optional (true, false). Default is false.

The following code sample is the toolbar configuration of the shipped product.

```
<toolbar>
  <button id="metadataPanel_addScope" iconCls="increase-scope-
icon"/>
  <button id="metadataPanel_reduceScope" iconCls="decrease-scope-
icon"/>
  <button id="metadataPanel_expandAll" iconCls="expand-all-icon"
align="right"/>
  <button id="metadataPanel_collapseAll" iconCls="collapse-all-
icon"/>
</toolbar
```

The following code sample modifies the shipped product behavior by removing the increase/decrease buttons and changing the alignment for the expand/collapse buttons.

```
<toolbar>
  <button id="metadataPanel_expandAll" iconCls="expand-all-icon"
align = "left" separator="true"/>
  <button id="metadataPanel_collapseAll" iconCls="collapse-all-
icon" align = "left" />
</toolbar>
```



Configuring meta elements

The `<meta-configs>` element contains a list of configurations for meta elements, to alter their default behavior in the **Properties** pane.

- `name`: Name of the meta element as defined in the rng schema file.
- `sort`: Sorts the list of values displayed in the panel for a meta element. Values are unsorted when no value is provided. Supported values are ASC and DESC.
    - ASC is ascending order (i.e. A to Z, 0 to 9). Earlier dates and times precede later ones.
    - DESC is descending order (i.e. Z to A, 9 to 0).
- `priority`: Specifies order of appearance for the meta element in the pane. A higher priority element appears before a lower priority element.

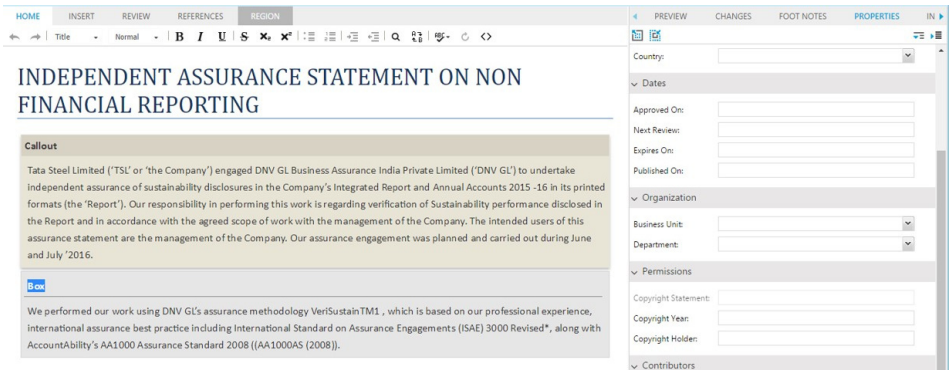
```
<meta-configs>
  <meta-config name="Meta Name" sort="ASC" priority ="10">
…
  </meta-config>
  <meta-config name="Another Meta" sort="ASC" priority ="10">
…
  </meta-config>
</meta-configs>
```

The `<display-filter>` element contains configuration options to alter the default display behavior like visibility and state.

- `readonly`: if set to **true**, disables edits to the field in the pane. Default value is false.
- `hidden`: if set to **true**, hides the field in the UI. Default value is false.
- `selector`: specifies rules to override the default behavior based on cursor location. The *readonly* and *hidden* behavior can be overridden for an element and optionally of a specific type. There can be multiple selectors in a `<display-config>` element with common behavior.
    - `element`: optional name of element on which meta is applied (for example region or table).
    - `elementType`: optional type of element on which meta is applied (for example box or callout for a region).
    - `section-type-path`: optional section hierarchy of the element (for example /document/section). Use this attribute to override the default behavior in a specific section only.

```
<meta-configs>
    <meta-config name ="copyrightStmt">
      <display-filter>
         <readonly>true</readonly>
      <display-filter>
    </meta-config>
```

```
    <meta-config name ="copyrightLicense">
      <display-filter>
          <hidden>true</hidden>
          <selector element="region" elementType="box">
      </display-filter>
    </meta-config>
</meta-configs>
```



- *data*: specifies the configuration for fetching data from an external system for a meta element.
- *providerref:* refers to the provider definition to get the values from the specified data provider
    - idref is used to refer to the id attribute of the data provider
- *valueField:* specifies the property name used to persist the selected value(s) in the XML. If valueField is not specified, then id will be used to store the selected value.
- *displayText*: specifies the property name used to display the value(s) in the pane. If displayText is not specified, then the text property will be used to display the selected value.

```
<meta-configs>
  <meta-config name="AttributeName" sort="ASC" priority ="10">
     <display-filter>
        <selector element="region" elementType="box" section-
type-path="/document/section"/>
    </display-filter>
    <data>
        <providerref idref="UserDomainProvider"/>
        <valueField>id</valueField>
        <displayText>text</displayText>
    </data>
  </meta-config>
</meta-configs>
<provider id="UserDomainProvider"
name="PlatformDomainValueProvider" handler="getDomainValues" >
            <param>Users</param>
            <param>true</param>
</provider>
```

The <provider> element provides configuration options for associating dynamic data and displaying such data for meta elements. PlatformDomainValueProvider is implemented as a sample data provider, but data could just as easily come from an external system. This element takes two parameters (<param>):

1. Attribute domain name used for fetching Platform values.

2. To specify to get values from cache or to always get the latest values from Platform. Set to **true** to always get the latest values from Platform. The default value is **false** to get data from cache.

```
<provider id="UserDomainProvider"
name="PlatformDomainValueProvider" handler="getDomainValues">
  <param>Users</param>
  <!-- Set to true to always get latest value from Platform. The
default value is false to get data from cache -->
  <!-- <param>true</param> -->
</provider>
```

Metadata can also be obtained from an external system by implementing a custom data provider. Essentially, the data provider is a Javascript function returning one or more JSON objects. The function name returning the JSON objects must appear in the handler attribute when adding an entry to the `metapane-config.xml` file. Here is an example which could be used to obtain values from a static JavaScript file:

```
<provider id="myDataProvider" name="externalDataProvider"
handler="getValues" >
</provider>
```

Quark recommends placing the static JavaScript file in this example at `{server}\webapps\workspace\editor\js\extension`.

Additional configuration steps are required:

1. Add the following entry to `dependent-files.xml` at `{server}\webapps\workspace\WEB-INF\classes`:

   ```
   <file type="javascript"
   path="editor/js/extension/CustomDataProvider.js"></file>
   ```

2. Associate the external data provider inside meta-config:

```
<meta-config name ="Users"  >
     <data>
          <providerref idref="myDataProvider" />
          <valueField>id</valueField>
          <displayText>text</displayText>
     </data>
</meta-config>
```

Obtaining dynamic data from an external system is also possible by making an Ajax request to an external system and providing appropriate exception/response handling. So long as the response loads data in JSON format and matches the pattern outlined above, poly-hierarchical domain values could be dynamically requested and surfaced inside the Quark Author experience.

Multi-channel preview

Configure Multi-channel preview by editing the `previewchannel-config.xml` file located at `{server}\webapps\workspace\editor\config\`.

Place this file under each content type folder to override the configuration settings based on content type.

The `Channels` element is used to specify the publishing channels for preview. For each content type specify the publishing channels that should be available to the user in the **Preview** tab of the assignment page.

- `displayName`: (Optional) Specifies the channel name displayed in the user interface.
- `id`: Specifies the Publishing Channel Id as defined in the Platform Server.
- `OutputFormat`: Supported values are:
    1. `IMAGE_ARCHIVE`: An image archive for the published output, which will be rendered inside a web page.
    2. `HTML_ARCHIVE`: An HTML archive for the published output, which will be rendered as pointing to the file name present in the HTML Archive.
    3. `PDF_ARCHIVE`: The published output PDF, which will be rendered as it is via PDF viewer plugins.
- `DownloadChannel`: (Optional) Used in case a different channel needs to be invoked for download of a selected channel preview.

```
<Channels>
  <Channel displayName="IMAGE" outputFormat="IMAGE_ARCHIVE"
id="smartDocJpeg"/>
  <Channel displayName="HTML" outputFormat="HTML_ARCHIVE"
id="smartDocHtml" downloadChannel="smartDocHtml"/>
  <Channel displayName="PDF" outputFormat="PDF_ARCHIVE"
id="smartDocPdf" downloadChannel="smartDocPdf"/>
</Channels>
```

Smart Document pane configuration

Configure the Smart Document pane by editing the `smartdoctree-config.xml` file located at `{server}\webapps\workspace\editor\config\`. This file is used to specify the context menu to be shown on instantiated sections and un-instantiated section nodes.

Add a new custom menu item by adding a menu entry under any of the menu groups. A default entry can also be removed to remove the corresponding context menu item.

- `id`: ID of the menu item.
- `text`: Default text of the menu.
- `icon`: URL for the menu icon
- `iconCLs`: CSS class for the menu icon.
- `override`: (Optional) **true** to override the default implementation for the menu item. Use when you need your own implementation to override the default implementation.

- `toolbar`: (Optional) **true** to also add the corresponding toolbar button.
- `align`: (Optional) Aligns the button in the toolbar if the toolbar option is **true**.
- `separator`: (Optional) If true, then adds a separator after the toolbar button corresponding to item.

```
 <instantiated-menus>
  <menu id="sectionView_ctxMenuItemCheckOut" text="Check Out"
iconCls="edit-component-icon" />
  <menu id="sectionView_ctxMenuItemSave" text="Save"
iconCls="save-component-icon"/>
  <menu id="sectionView_ctxMenuItemCancelCheckout" text="Discard
Changes" iconCls="discard-changes-icon"/>
  <menu id="sectionView_ctxMenuCreateCmp" text="Create Component"
iconCls="create-component-icon"/>
  <menu id="sectionView_ctxRefreshComp" text="Refresh Component"
iconCls="refresh-component-icon"/>
</instantiated-menus>
<uninstantiated-menus>
  <menu id="sectionView_ctxMenuFromServer" text="Create from
Server" iconCls="component-from-server-icon"/>
  <menu id="custom_ctxImportFromUrl" text="Create from URI"
icon="images/create-from-server.png"/>
</ uninstantiated-menus>
```

Asset browser configuration

The `AssetBrowserSettings` element is used to specify the settings for the asset picker dialog used for browsing and inserting components, images, Excel data or smart tables from the editor.

The following is the set of current `AssetBrowserSettings` shipped with the product:

- `SMARTDOC_BROWSER`
- `SMARTSECTION_BROWSER`
- `PICTURE_BROWSER`
- `VIDEO_BROWSER`
- `DATATABLE_BROWSER`
- `REGION_BROWSER`
- `EXCEL_TABLE_BROWSER`
- `EXCEL_CHART_BROWSER`
- `PPT_BROWSER`
- `VISIO_BROWSER`
- `QUARKCOPYDESK_BROWSER`
- `QUARKXPRESSPROJECT_BROWSER`

To configure the asset picker dialog in a way that only PNG images with a status of `Approved` can be inserted in the Smart Document:

1. Update the `Workspace-Config.xml` file located at `{server}\webapps\workspace\WEB-INF\classes\`, to create new browser

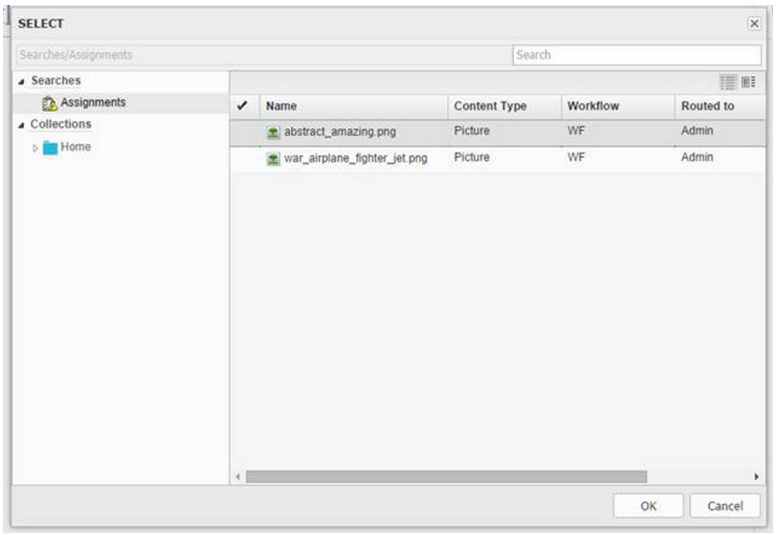
settings that search for content type **Picture** and apply the additional filters `Status=Approved; File extension=png.`

```
<AssetBrowserSetting id="PICTURE_BROWSER_PNG"
  searchForContentType="Picture"
  includeChildContentTypes="true"
  filter="Status=Approved;
  File extension=png"/>
```

2. Update the `xmleditor-config.xml` file for “Smart Document” content type to use the new settings while selecting an image from the asset browser.

```
<assetpicker-settings>
  <assetpicker-setting id="PICTURE_BROWSER"
assetBrowserId="PICTURE_BROWSER_PNG"/>
</assetpicker-settings>
```

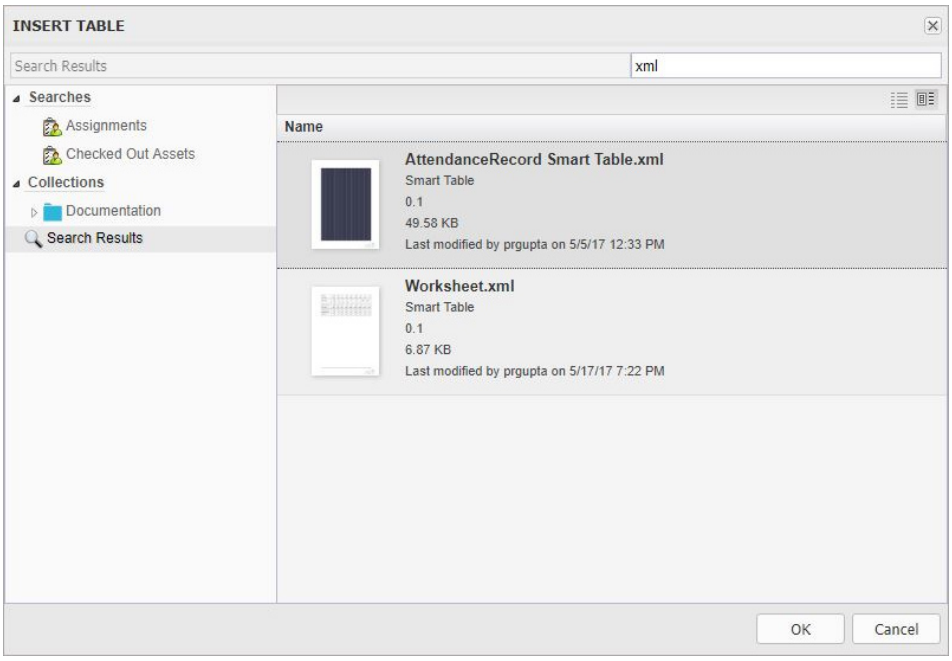
Only PNG images with a status of `Approved` will be shown in the asset browser dialog.



Region or Table components

Asset browsers corresponding to region or table components are configured using the `blockpicker-settings` element. Each < blockpicker-setting> defined in `xmleditor-config.xml` maps a content type (corresponding to block-level XML documents) to the asset browser ID defined in `Workspace-Config.xml`.

```
<blockpicker-settings>
    <blockpicker-setting contentType="Smart Region"
assetBrowserId =                    "REGION_BROWSER" />
    <blockpicker-setting contentType="Smart Table" assetBrowserId
=                   "DATATABLE_BROWSER" />
</blockpicker-settings>
```



Section Picker configuration

The `sectionpicker-settings` element is used to specify the settings for the asset picker dialog used for browsing and inserting sections from the editor.

To configure the asset picker dialog in a way that only Smart Section with a status of Approved can be inserted in the Smart Document:

1. Update the `Workspace-Config.xml` file to create new browser settings that searches for the "Smart Section" content type, and apply the following additional filter: `Status=Approved`.

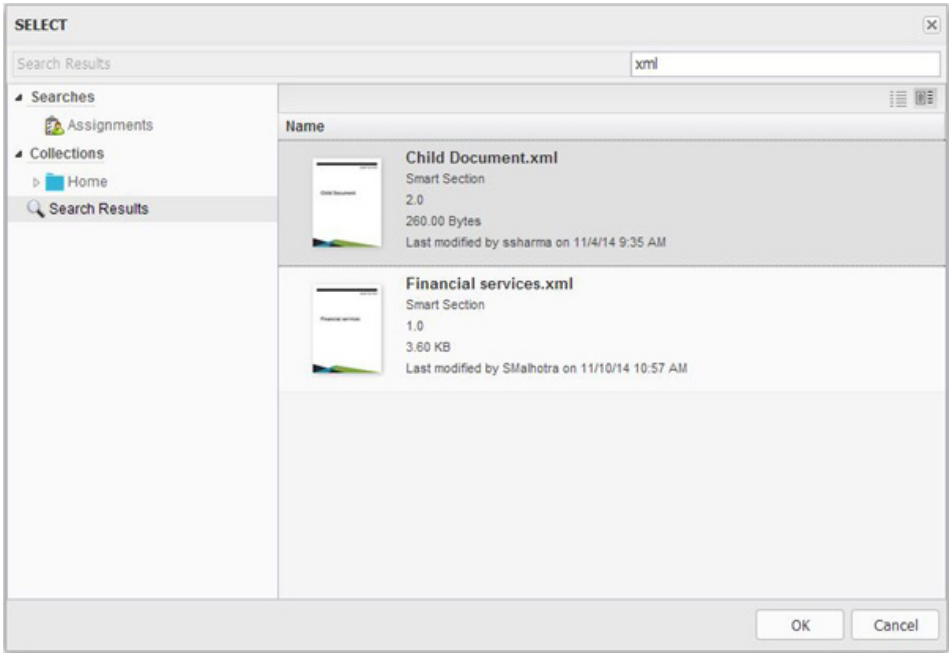
   ```
   <AssetBrowserSetting id="SMART_SECTION_BROWSER"
   searchForContentType="Smart Section"
   includeChildContentTypes="true"
   filter="Status=Approved"/>
   ```

2. Update the `xmleditor-config.xml` file so that the "Smart Document" content type uses any existing settings while selecting an asset corresponding to a section specific content type.

   ```
   <sectionpicker-settings>
     <sectionpicker-setting contentType="Smart Section"
   assetBrowserId="SMART_SECTION_BROWSER"/>
   </sectionpicker-settings>
   ```

   The `contentType` value is used to determine the asset picker settings to be used for a given section type. The value of `contentType` should match the Platform content type.

Only Smart Sections with a status of `Approved` will be shown in the asset browser dialog.



Header configuration

The `headertoolbar` element is used to configure the Editor's page toolbar and the buttons per document type.

- `showheader`: Use to specify whether the toolbar and its buttons should be hidden.
- `headerlogo`: Specifies the path to the banner image. The default banner displayed in the editor can be changed by updating the value of this attribute.

Buttons can be added to the toolbar by adding the following entry under the `headertoolbar` element.

```
<button view-name="hdr-tbar-btn-save-locally"
text="Save Locally"
icon="/images/save-local.png"
mode="edit"/>
```

- `lowResHeaderlogo`: Specifies the path to a low-resolution version of the banner image.
- `view-name`: Specifies the Id of the button.
- `text`: Specifies the label for the button.
- `iconCls`: Specifies the path to the icon for the button.
- `mode`: Specifies the document mode supported by the button. The value edit disables the button if the document is opened in read-only mode.

Extjs based controller classes can be loaded by adding them to the namespace settings.

```
<namespaces>
  <namespace name="com.quark.kestrel.feature" path="js/feature">

<controller>com.quark.kestrel.feature.controller.CheckInDlgContro
ller</controller>
  </namespace>
```

```
  <namespace name="com.quark.kestrel.extension"
path="js/extension"/>
</namespaces>
```

Application settings

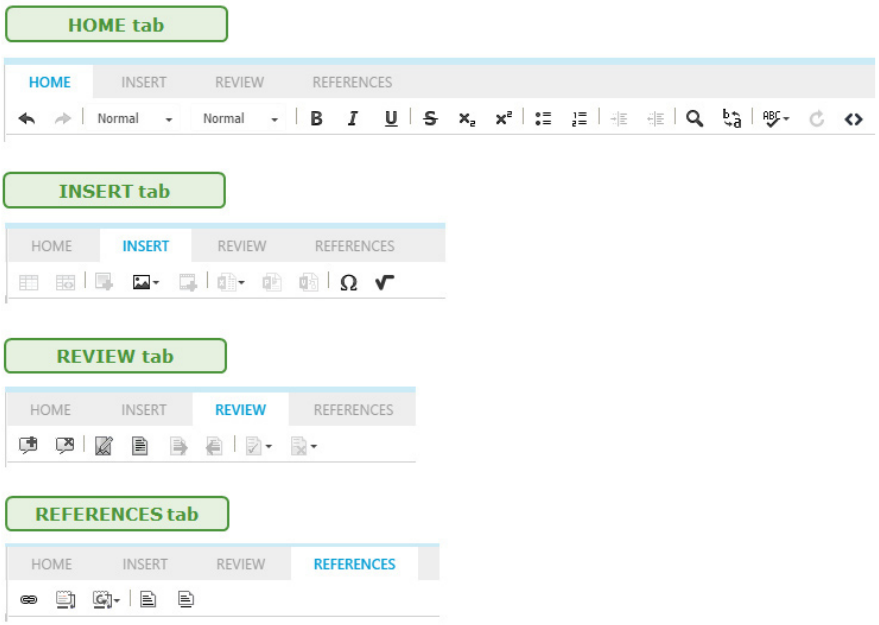
General application level settings are configured using the `application-settings` element and can be specified per document type.

- `ajaxTimeout`: Specifies the timeout in milliseconds for all Ajax requests.
- `showRevisionSettings`: Specifies whether to show previous revision comments on document checkout .
- `allowPublishedRenditionDownload`: Specifies whether to allow download of the published rendition of an asset .
- `showAttributeForm`: Specifies whether to show the attribute form view in the **Check In** dialog. Set to `False` to hide the form view.

```
<application-settings>
  <add key="ajaxTimeout" value="300000"/>
  <add key="showRevisionSettings" value="true"/>
  <add key="allowPublishedRenditionDownload" value="true"/>
  <add key="showAttributeForm" value="true"/>
</application-settings>
```

Toolbar tabs configuration

Quark Author Web Edition utilizes a tab-based toolbar for the core editing tools. The tabs and buttons are rendered according to the configuration. Define the tab-based toolbar to hide the native Editor toolbar. The configuration for the toolbar corresponds to the content
types.



The tabs are defined in the resource files (`{content_type}-resource_{lang}.xml`) specific to the content type found here: `{server}\webapps\workspace\editor\config\{content type}\`.

```
<resources>
  <toolbar>
    <tab id="home" label="Home"></tab>
    <tab id="insert" label="Insert"></tab>
    <tab id="references" label="References"></tab>
    <tab id="review" label="Review"></tab>
  </toolbar>
</resources>
```

The toolbar tab configuration is in the `xmleditor-config.xml` file located in the `{server}\webapps\workspace\editor\config\` folder. This is optional. If toolbar configuration is not provided, then the UI will switch to the traditional toolbar.

```
<configuration>
  ........
  <toolbar>
    <tab id="review" type="main">
      <items>
        <item>addComment</item>
        <item>deleteComment</item>
        <item>-</item>
        <item>Tracking</item>
        <item>Highlighting</item>
        <item>NextChange</item>
        <item>PreviousChange</item>
        <item>-</item>
        <item>Accept</item>
        <item>Reject</item>
      </items>
    </tab>
    <tab id="references" type="main" mode="edit">
      <items>
        <item>Link</item>
        <item>xref</item>
        <item>updatexref</item>
        <item>-</item>
        <item>FootNote</item>
        <item>EndNote</item>
      </items>
    </tab>
  </toolbar>
  ........
</configuration>
```

- The `tab` element is used to represent the toolbar tabs.
- Each button/dropdown to be displayed when the tab is selected is represented by the `item` element.

➡ A button/ richCombo may be included in more than one toolbar tab.

Custom toolbar tabs

You can define custom tabs to add to the toolbar.

To configure a custom toolbar tab:

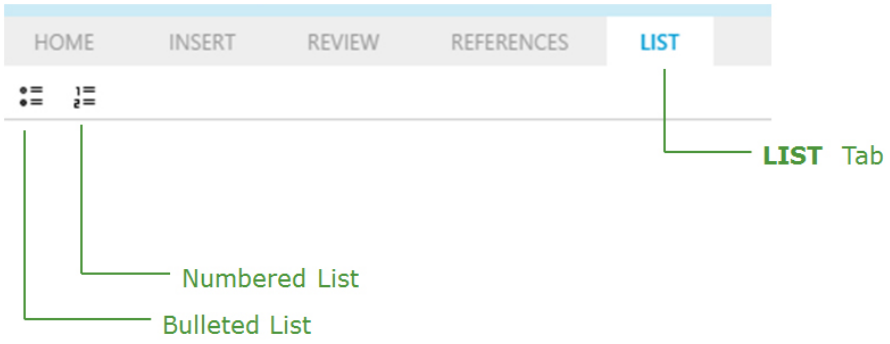
1. Add the toolbar buttons in the `xmleditor-config.xml` file located in the `{server}\webapps\workspace\editor\config\` folder.

```
<toolbar>
  <tab id="List" type="main">
    <items>
      <item>BulletedList</item>
```

```
      <item>NumberedList</item>
    </items>
  </tab>
</toolbar>
```

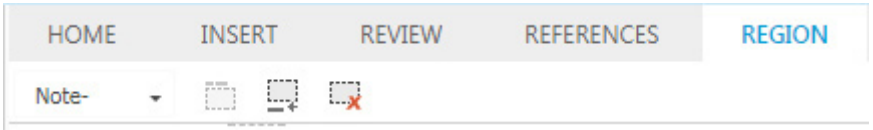
2. Provide the text for the new tab in the resource files (`{content_type}-resource_{lang}.xml`) specific to the content type found here: `{server}\webapps\workspace\editor\config\{content type}\`.

```
<resources>
  <toolbar>
    <tab id="home" label="Home"></tab>
    <tab id="insert" label="Insert"></tab>
    <tab id="references" label="References"></tab>
    <tab id="review" label="Review"></tab>
    <tab id="list" label="List"></tab>
  </toolbar>
</resources>
```



Dynamic toolbar tabs

Quark Author Web Edition allows for dynamic, contextual toolbar tabs which appear only when the cursor location is in a specific document location (e.g. region, table, figure, etc.). Buttons relevant to current context are shown under dynamic tabs. Tabs for **Table** and **Region** elements are part of the default configuration.



The **REGION** tab appears when a user inserts a new region or starts editing an existing region.

The **TABLE** tab appears when a user inserts a new table or starts editing an existing table.

You can define custom dynamic tabs to add to the toolbar. A toolbar tab of type `toolTab` specifies a contextual toolbar. Making the toolbar tab appear according to cursor location is configured using the `selector` element.

To configure a dynamic toolbar tab:

1. Provide the text for the new tab and add the toolbar buttons in the `xmleditor-config.xml` file located in the `{server}\webapps\workspace\editor\config\` folder.

   *To configure a dynamic tooltab called **TABLE** that only appears when the cursor is inside a table:*

```
<toolbar>
  <tab id="table" type="toolTab" mode="edit">
    <selector element-name="table"/>
    <selector element-name="bodydiv" element-
type="formaltable"/>
    <items>
      <item>tableTypeCombo</item>
      <item>insert-table-title</item>
      <item>insert-table-description</item>
      <item>add-para-after</item>
      <item>-</item>
      <item>insert-row-before</item>
      <item>insert-row-after</item>
      <item>insert-column-before</item>
      <item>insert-column-after</item>
      <item>-</item>
      <item>split-cells-horizontal</item>
      <item>split-cells-vertical</item>
      <item>merge-cells</item>
      <item>-</item>
      <item>row-delete</item>
      <item>column-delete</item>
      <item>delete-table</item>
    </items>
  </tab>
</toolbar>
```

   *To configure a dynamic tooltab called **REGION** that only appears when the cursor is inside a region:*

```
<tab id="region" type="toolTab" mode="edit">
  <selector element-name="region"/>
  <items>
    <item>regionTypeCombo</item>
    <item>insert-region-title</item>
    <item>insert-para-after-region</item>
    <item>remove-region</item>
  </items>
</tab>
```

   You can add an element-type attribute to the selector element, adding dynamic behavior for a specific element.

   *To force focus to the **REGION** tab only when a region of type Box is created:*

```
<tab id="region" type="toolTab" mode="edit">
  <selector element-name="region" element-type="box"/>
  <items>
    <item>regionTypeCombo</item>
    <item>insert-region-title</item>
    <item>insert-para-after-region</item>
    <item>remove-region</item>
  </items>
</tab>
```

Editor configuration

Use the `editor-config.xml` file located in the `{server}\webapps\workspace\editor\js\thirdparty\ckeditor\` folder to specify the Editor toolbar configuration. Place this file under each content type folder to override the configuration settings based on content type.

Configure this file to specify the following:

- Keyboard Shortcuts configuration
- Default settings for track changes
- Plugins Configuration
- Reference note configuration
- Cross reference configuration
- Copy/Paste behavior configuration
- Section component check in configuration

Supports overriding of selective configuration elements which means if other elements are not specified then the value will be picked from the file in this location.

Keyboard shortcuts configuration

The `keystrokes` element is used to configure keyboard shortcuts:

```
 config.keystrokes = [
  [CKEDITOR.CTRL + QXmlEditorConstants.keys.B, 'bold'],
  [CKEDITOR.CTRL + QXmlEditorConstants.keys.I, 'italic'],
  [CKEDITOR.CTRL + QXmlEditorConstants.keys.U, 'underline'],
  [CKEDITOR.ALT + CKEDITOR.SHIFT + QXmlEditorConstants.keys.F,
QXmlEditorConstants.commands.FIND],
  [CKEDITOR.ALT + CKEDITOR.SHIFT + QXmlEditorConstants.keys.H,
QXmlEditorConstants.commands.REPLACE],
  [CKEDITOR.ALT + CKEDITOR.SHIFT + QXmlEditorConstants.keys.L,
QXmlEditorConstants.commands.INSERT_LINK],
  [CKEDITOR.ALT + CKEDITOR.SHIFT + QXmlEditorConstants.keys.T,
QXmlEditorConstants.commands.INSERT_TABLE],
  [CKEDITOR.ALT + CKEDITOR.SHIFT + QXmlEditorConstants.keys.C,
QXmlEditorConstants.commands.ADD_COMMENT],
  [CKEDITOR.ALT + CKEDITOR.SHIFT + QXmlEditorConstants.keys.I,
QXmlEditorConstants.commands.INSERT_IMAGE],
  [CKEDITOR.ALT + CKEDITOR.SHIFT + QXmlEditorConstants.keys.G,
QXmlEditorConstants.commands.INSERT_FIGURE],
  [CKEDITOR.ALT + CKEDITOR.SHIFT + QXmlEditorConstants.keys.E,
QXmlEditorConstants.commands.ENABLE_TRACKING],
  [CKEDITOR.ALT + CKEDITOR.SHIFT + QXmlEditorConstants.keys.A,
```

```
QXmlEditorConstants.commands.ACCEPT_CHANGE],
  [CKEDITOR.ALT + CKEDITOR.SHIFT + QXmlEditorConstants.keys.R,
QXmlEditorConstants.commands.REJECT_CHANGE],
  [CKEDITOR.ALT + CKEDITOR.SHIFT + QXmlEditorConstants.keys.V,
QXmlEditorConstants.commands.INSERT_VIDEO],
  [CKEDITOR.ALT + CKEDITOR.SHIFT + QXmlEditorConstants.keys.N,
QXmlEditorConstants.commands.APPLY_NORMAL_INLINE_STYLE],
  [CKEDITOR.ALT + CKEDITOR.SHIFT + QXmlEditorConstants.keys.U,
QXmlEditorConstants.commands.BULLETED_LIST],
  [CKEDITOR.ALT + CKEDITOR.SHIFT + QXmlEditorConstants.keys.O,
QXmlEditorConstants.commands.NUMBERED_LIST],
  [CKEDITOR.CTRL + QXmlEditorConstants.keys.TWO,
QXmlEditorConstants.commands.OPEN_INLINE_STYLE_COMBO],
  [CKEDITOR.CTRL + QXmlEditorConstants.keys.ONE,
QXmlEditorConstants.commands.OPEN_PARA_STYLE_COMBO],
  [CKEDITOR.CTRL + QXmlEditorConstants.keys.S,
QXmlEditorConstants.commands.SAVE_REVISION],
  [CKEDITOR.CTRL + CKEDITOR.SHIFT + QXmlEditorConstants.keys.S,
QXmlEditorConstants.commands.SAVE_AND_CLOSE],
  [CKEDITOR.CTRL + CKEDITOR.ALT + QXmlEditorConstants.keys.Q,
QXmlEditorConstants.commands.ABORT_DOCUMENT],
  [CKEDITOR.CTRL + QXmlEditorConstants.keys.Q,
QXmlEditorConstants.commands.CLOSE_DOCUMENT]
  [CKEDITOR.ALT + CKEDITOR.SHIFT + QXmlEditorConstants.keys.Q,
QXmlEditorConstants.commands.INSERT_EQUATION],
  [CKEDITOR.ALT + QXmlEditorConstants.keys.RIGHT_ARROW,
QXmlEditorConstants.commands.SELECT_NEXT_RIGHT_TAB_PANE],
  [CKEDITOR.ALT + QXmlEditorConstants.keys.LEFT_ARROW,
QXmlEditorConstants.commands.SELECT_PREVIOUS_RIGHT_TAB_PANE],
  [CKEDITOR.ALT + QXmlEditorConstants.keys.RIGHT_ANGULAR,
QXmlEditorConstants.commands.SELECT_NEXT_LEFT_TAB_PANE],
  [CKEDITOR.ALT + QXmlEditorConstants.keys.LEFT_ANGULAR,
QXmlEditorConstants.commands.SELECT_PREVIOUS_LEFT_TAB_PANE],
  [CKEDITOR.ALT + QXmlEditorConstants.keys.RIGHT_BRACKET,
QXmlEditorConstants.commands.SELECT_NEXT_EDITOR_TAB],
 [CKEDITOR.ALT + QXmlEditorConstants.keys.LEFT_BRACKET,
QXmlEditorConstants.commands.SELECT_PREVIOUS_EDITOR_TAB],
  [CKEDITOR.ALT + QXmlEditorConstants.keys.H,
QXmlEditorConstants.commands.SELECT_EDITOR_HOME_TAB],
  [CKEDITOR.ALT +
QXmlEditorConstants.keys.I,QXmlEditorConstants.commands.SELECT_ED
ITOR_INSERT_TAB],
  [CKEDITOR.ALT + QXmlEditorConstants.keys.R,
QXmlEditorConstants.commands.SELECT_EDITOR_REVIEW_TAB],
  [CKEDITOR.ALT + QXmlEditorConstants.keys.E,
QXmlEditorConstants.commands.SELECT_EDITOR_REFERENCES_TAB]
];
```

Shortcuts can also be added for the following commands:

- INSERT_EXCEL_TABLE
- INSERT_EXCEL_CHART
- INSERT_DATATABLE
- HIGHLIGHTING
- ACCEPT_ALL_CHANGES
- REJECT_ALL_CHANGES
- NEXT_CHANGE
- PREVIOUS_CHANGE
- DELETE_COMMENT

- VIEW_XML
- INSERT_LINK
- APPLY_NORMAL_PARA_STYLE
- INSERT_FOOTNOTE
- INSERT_ENDNOTE
- INSERT_XREF
- UPDATE_XREF
- UPDATE_ALL_XREF

Track changes and plugins configuration

The `tracking` element in the `editor-config.js` file located in the `{server}\webapps\workspace\editor\js\thirdparty\ckeditor` folder is used to configure the change tracking feature.

- `useDynamicColors`: Use to specify that you want to use automatic colors to show tracked content.
- `autoStartup`: Use to enable tracking by default for all documents.
- `defaultInsertionColor`: Use to specify the default insertion color for the logged in user when `tracking_useDynamicColors` is `true`.
- `defaultDeletionColor`: Use to specify the default deletion color for the logged in user when `tracking_useDynamicColors` is `true`.
- `dynamicColors`: Use to specify a default list of dynamic colors used when `tracking_useDynamicColors` is `true`.
- `extraPlugins`: Use to add new toolbar plugins or remove unused plugins
- `toolbar_XmlEditor`: Use to configure how to show toolbar buttons in different groups.

```
config.tracking_useDynamicColors = true;
config.tracking_defaultInsertionColor = "#0000FF";
config.tracking_defaultDeletionColor = "#FF0000";
config.tracking_dynamicColors = [ '#996600', '#83CEFF',
'#00A600', '#CC0099', '#E68A00', '#AD85FF', '#62A382', '#B26B47',
'#00CCA3', '#7ACC29', '#008F6B', '#706680', '#660066', '#B28F8F',
'#FF99CC', '#990000', '#751975', '#006B6B', '#85C2A3', '#99B2FF'
];
```

Reference note configuration

Reference notes type configuration

The `note_suportedTypes` element is used to specify the list of note types that are supported and displayed in the editor.

- `type`: Specifies the type of reference note (footnote or endnote).
- `displayName`: Specifies the text to be displayed in the reference note.

```
config.note_supportedTypes =
  [{type: 'footnote',
    displayName: 'Footnote' },
   {type: 'endnote',
    displayName: 'Endnote:}
  ];
```

Reference notes styling configuration

Style for the reference notes on the canvas can be configured in the `contents.css` file found here:
`{server}\webapps\workspace\editor\js\thirdparty\ckeditor\`

```
 body
  {counter-reset: footnote endnote;}
  .footnote
  {counter-increment: footnote;}
  .endnote
  {counter-increment: endnote;}
  .footnote:before
  {content: counter(footnote,lower-roman);}    // Any list-style
type can be used here
  .endnote:before
  {content: counter(endnote, decimal);}       //  Any list-style
type can be used here
```

Style for the reference notes in the footnotes pane can be configured in the `xml-editor-styles.css` file found here:
`{server}\webapps\workspace\editor\css`

```
.metro-tree- panel,.note -view-panel.x-component
  {counter-reset: footnote endnote;}
  .footnote
  {counter-increment: footnote;}
  .endnote
  {counter-increment: endnote;}
  .footnote:before
  {content: counter(footnote,armenian);}      // Any list-style
type can be used here
  .endnote:before
  {content: counter(endnote,upper-alpha);}    // Any list-style
type can be used here
```

Reference notes metadata configuration

Use the `Smart Document.rng` file located in the `{server}\webapps\workspace\editor\config\Smart Document` folder to specify metadata for reference notes. Metadata properties can be defined using '`refnote-meta`'.

Use the `common-tags.rng` file located in the `{server}\webapps\workspace\editor\config\Smart Section` folder to specify metadata for reference notes included inside tagged content.

Smart Document.rng

```
<define name="content-model">
  <zeroOrMore>
    <parentRef name="section"/>
  </zeroOrMore>
  <zeroOrMore>
    <parentRef name="appendix"/>
  </zeroOrMore>
</define>
```

```
<define name="refnote-meta">
  <parentRef name="audience"/>
  <parentRef name="permissions"/>
</define>
```

common-tags.rng

```
<define name="keyword">
   <grammar>
     <include href="Smart-Tag.rng">
       <define name="tag-type">
         <value>keyword</value>
       </define>
       <define name="nested-tags">
         <parentref name="all.section-tags"/>
       </define>
       <define name="tag-meta">
         <parentref name="audience"/>
         <parentref name="permissions"/>
       </define>
       <define name="refnote-type">
         <parentref name="common-refnote-types"/>
       </define>
       <define name="refnote-meta">
                    <parentRef name="audience"/>
                    <parentRef name="permissions"/>
       </define>
     </include>
   </grammar>
 </define>
```

Cross references configuration

The `referred_criteria` element is used to specify cross references types and expression value for displaying text.

- `target_type`: Specifies the cross reference type (section, table or figure).
- `default_text`: Default text to be shown in case no text exists at the target element or the selector element doesn't exist for the reference type
- `expressions`: Specifies the relative xpath of the element from which the text is to be extracted for displaying reference text.

```
config.referred_criteria =
  [{target_type: 'section',
      expressions:[{selector:'title'}],
      default_text:'Section Reference'},
   {target_type:'table',
      expressions:[{selector:'title'},
                   {selector:'desc'},
                   {selector:'p[@type = table-title]'},
                   {selector:'p[@type = table-desc]'}],
      default_text:'Table Reference'},
   {target_type:'figure',
      expressions:[{selector: 'p'}],
      default_text:'Figure Reference'},
   {target_type:'defaultRegionType',
      expressions:[{selector:'p[1]', maxCharacters:32}],
      default_text:'Reference'},
   {target_type:'box',
      expressions:[{selector:'p[1]', maxCharacters:32}],
      default_text:'Box Reference'},
   {target_type:'callout',
      expressions:[{selector:'p[1]', maxCharacters:32}],
```

```
default_text:'Callout Reference'}];
```

Hiding cross-reference target elements

To configure a list of reference types to be displayed in the **INSERT CROSS REFERENCE** dialog:

1. Open the `editor-config.js` file located at `{server}\webapps\workspace\editor\js\thirdparty\ckeditor.`
2. Locate the option by searching for the string **config.referred_targets**.
3. Specify the list of targets to display in the dialog.

   Examples:

To display all regions of type 'box' in the dialog (irrespective of its location in the document):

```
config.referred_targets = [
  {
    element: 'region',
    elementType : 'box'
  }
];
```

To display only box type regions which are in the appendix section:

```
config.referred_targets = [
  {
    element: 'region',
    elementType : 'box'
  }
];
```

Preventing editing of cross-reference text for target elements

To prevent the editing of of cross-reference text of target elements in the **INSERT CROSS REFERENCE** dialog:

1. Open the `editor-config.js` file located at `{server}\webapps\workspace\editor\js\thirdparty\ckeditor.`
2. Locate the option by searching for the string **config.referred_criteria**.
3. Set the `readOnly` property to **true**.

```
config.referred_criteria = [
{
  target_type: 'section',
  expressions: [
  {
    selector :'title'
  }],
  readOnly : true,
  default_text: 'Section Reference'
}];
```

Copy/Paste behavior configuration

To prevent the copy/paste of metadata:

1. Open the `editor-config.js` file located in the `{server}\webapps\workspace\editor\js\thirdparty\ckeditor` folder.
2. Set the following property to false:

```
config.clipboard_enableCopyOfMetadata = false;
```

To prevent the copy/paste or drag/drop of images on the canvas:

1. Open the `editor-config.js` file located in the `{server}\webapps\workspace\editor\js\thirdparty\ckeditor` folder.
2. Set the following property to false:

```
config.clipboard_enablePastingOfImage = false;
```

Saving components silently

To allow a user to save section components without invoking a checkin dialog box:

1. Open the `Workspace-Config.xml` file located in the `{server}\webapps\workspace\editor\js\thirdparty\ckeditor` folder.
2. Set the `SaveSilenty` property to true:

```
<RevisionSetting operation="SaveComponent">
   <SaveSilently>true</SaveSilently>
   <SaveAsMinorVersion>false</SaveAsMinorVersion>
</RevisionSetting>
```

➡ Setting the property to false will force the **CHECKIN** dialog to appear when sections have been edited in a Smart Document.

Atomic Elements

Specify list of block types that should be copied as whole even if partial content is selected.

Partial selection implies that selection span across at least a block.

Currently, only block element of type bodydiv is supported.

```
config.clipboard_atomicElements = [
        /*{
            element : 'bodydiv',
            elementType : 'figure'
        }*/
    ]
```

DOMOptimization for large tables

Allow optimization of DOM by collapsing content of referenced block elements

config.reference_enableDOMOptimization = true;

Custom region configuration

Use the `Smart Document.rng` file located in the `{server}\webapps\workspace\editor\config\Smart Document` folder to define a new custom region.

```
<define name="region-types">
<choice>
      <value>callout</value>
      <value>box</value>
      <value>exhibit</value>
</choice>
</define>
<define name="region-title-type">
      <value>region-title</value>
</define>
```

New Region Type – "Exhibit"

When defining custom regions, tables, sections or figures, do not use a Smart Content source element semantic as a type value for another element. For example, do not define these in the following ways:

- <region type="section">
- <tag type="region">
- <section type="tag">

Specify this new region in the following files:

- The `Smart Document-Resource_en.xml` resource file located in the `{server}\webapps\workspace\editor\config\Smart Document` folder.

```
<!-- Regions -->
      <region type="box" label="Box"/>
      <region type="callout" label="Callout"/>
      <region type="exhibit" label="Exhibit"/>
```

Entry for New Region Type – "Exhibit" in Resources file

- Define the css for the newly created region type in the `editor-content.css` file located in the `{server}\webapps\workspace\editor\config\Smart Document` folder.

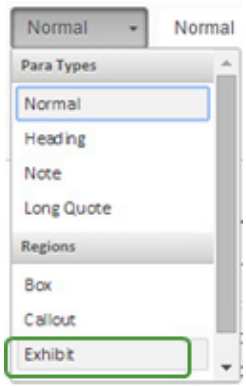
```
.exhibit {
   background-color: #CCCC99;
       border: 1px solid #D6D6D6;
}
```

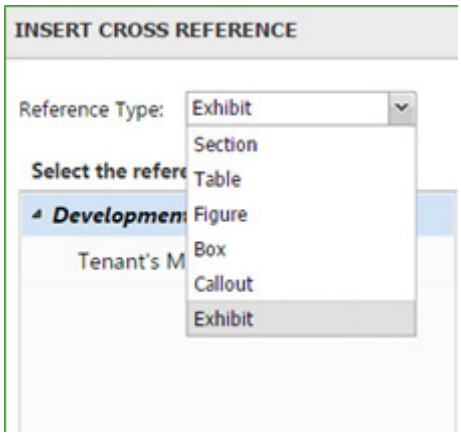
Defining css for the newly created region type "Exhibit"

- Define a cross reference type for the newly created region in the `editor-config.js` file located in the `{server}\webapps\workspace\editor\js\thirdparty\ckeditor` folder.

```
{
target_type : 'exhibit',
expressions : [ {
        selector : 'p[1]',
        maxCharacters : 32
} ],
default_text : 'Exhibit Reference'
}
```

The new region will be available as a resource and a reference:





Hiding regions

Region types can be removed completely from the Para Types combo box using the `hideRegionGroup` element.

1. Open the `editor-config.js` file located in the `{server}\webapps\workspace\editor\js\thirdparty\ckeditor` folder.
2. Locate the option by searching for the string “hideRegionGroup”. Set to true to hide the regions types from the Para Types combo box.

   ```
   config.region_hideRegionGroup = true;
   ```

Math equation configuration

Integration with the WIRIS editor enables the editor to insert and edit mathematical formulas in Quark Author.

By default this plugin is disabled.

➡ Enabling this feature requires a license key, which can be obtained from Quark Alliance. The license key is not shipped with the software.

To enable this plugin:

1. Open the `xmleditor-config.xml` file located in the `{server}\webapps\workspace\editor\config` folder.
2. Add an entry for the Wiris formula editor as shown below:

```
<tab id="insert" type="main" mode="edit"
  <items>
    <item>calstable</item>
    <item>datatable</item>
    <item>insert-region</item>
    <item>insert-region-from-server</item>
    <item)-</item>
    ....
    <item)-</item>
    <item>SpecialChar</item>
    <item>ckeditor_wiris_formulaEditor</item>
  </items>
</tab>
```

3. Open the `editor-config.js` file located at `{server}\webapps\workspace\editor\js\thirdparty\ckeditor` and uncomment `//config.extraPlugins = 'ckeditor_wiris';`

➡ If the settings in the `xmleditor-config.xml` file have been customized for different document types then the above change has to be done there as well. Document type specific `xmleditor-config.xml` files are located at `{server}\webapps\workspace\editor\config\<Document type>.`

4. Configure the Wiris license key. *(Work with Quark Alliance to obtain a license key.)*

   Open the `web.xml` file located at `{server}\webapps\pluginwiris_engine\WEB-INF` and add the license key in the `<param-value>` tag.

```
<description>WIRIS editor</description>
<context-param>
    <description>Here goes the product key of the
license.</description>
    <param-value></param-value>
</context-param>
```

There are various properties related to the output format of equations added to the canvas that can be configured. Review the `configuration.ini` file located at `{server}\webapps\pluginwiris_engine\WEB-INF\pluginwiris`.

➡ For more information on configuring the WIRIS editor, please visit *www.wiris.com/plugins/docs/resources/configuration-table*

Smart content editor reusable URLs

The editor page can be opened by passing the `user name`, `password` and `assetId` as request parameters in the following format in a browser:

```
http://<server>:<port>/workspace/login.qsp?userName=<userName>&Pa
ssword=<password>
&redirectUrl=/workspace/checkoutXMLDocument.qsp?assetId=<assetId>
```

The editor page can be opened in read-only mode by additionally specifying the parameter `mode` with a value of **readonly**:

```
http://<server>:<port>/workspace/login.qsp?userName=<userName>&Pa
ssword=<password>
&redirectUrl=/workspace/checkoutXMLDocument.qsp?assetId=<assetId>
%26mode=readOnly
```

The editor page can be opened in read-only mode for previous versions of document by additionally specifying the parameters `majorVersion` and `minorVersion` with a value of the **version number** of the desired version of the document:

```
http://<server>:<port>/workspace/login.qsp?userName=<userName>&Pa
ssword=<password>
&redirectUrl=/workspace/checkoutXMLDocument.qsp?assetId=<assetId>
%26mode=readOnly
%26majorVersion=<major version number>%26minorVersion=<minor
version number>
```

The checkout URL can be further customized to collapse the panes on the left- or right-hand side when the editor is loaded. The following URL collapses both the left and the right-hand panes:

```
http://<server>:<port>/workspace/checkoutXMLDocument.qsp?assetId=
<assetId>&collapseLeftPanel=true
&collapseRightPanel=true
```

The panes can be completely hidden using the following URL:

```
http://<server>:<port>/workspace/checkoutXMLDocument.qsp?assetId=
<assetId>&hideLeftPanel=true
&hideRightPanel=true
```

The editor page can be launched to create a new document of a specific platform content using the following URL:

```
 http://<server>:<port>/workspace/createNewXMLDocument
qsp?contentType=Smart Document
```

The editor page can also be launched to create a new document from an existing document:

```
 http://<server>:<port >/workspace/createNewXMLDocument
FromTemplate.qsp?assetId=<assetId>
```

Legal notices